



OPEN Project

STREP Project FP7-ICT-2007-1 N.216552

Title of Document: Initial prototype applications

Editor(s): Giancarlo Cherchi, Francesca Mureddu

Affiliation(s): Arcadia Design

Contributor(s): Holger Klus, Carmen Santoro, Susan Thomas, Joerg Doerflinger

Affiliation(s): Clausthal University, ISTI-CNR, SAP AG

Date of Document: January, 2009

OPEN Document: D5.2

Distribution: EU

Keyword List: Applications, Prototypes, Instructions, Usage, Design, Software, Hardware, Requirements

Version: 1.0

OPEN Partners:

CNR-ISTI (Italy)
Aalborg University (Denmark)
Arcadia Design (Italy)
NEC (United Kingdom)
SAP AG (Germany)
Vodafone Omnitel NV (Italy)
Clausthal University (Germany)

"The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The above referenced consortium members shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials subject to any liability which is mandatory due to applicable law. Copyright 2008 by Arcadia Design, Clausthal, CNR, Vodafone."

Title: Initial application prototypes	Id Number: WP5 – D5.2
--	------------------------------

Abstract

This document contains the description of the software prototypes delivered for D5.2 in terms of hardware and software requirements, setup and run, and usage instructions, needed for their correct operation. In particular, for the Business Domain, a first implementation of the Emergency Scenario presented in D5.1 is described, whereas for the Game Domain, three different prototypes are presented, namely Web PacMan, Java PacMan, and Social Game, respectively.

Title: Initial application prototypes	Id Number: WP5 – D5.2
--	------------------------------

Table of Contents

1. INTRODUCTION.....	2
2. BUSINESS DOMAIN PROTOTYPE.....	3
2.1. HARDWARE AND SOFTWARE REQUIREMENTS	3
2.2. SETUP AND RUN.....	3
2.3. USAGE.....	4
3. GAME DOMAIN PROTOTYPES	7
3.1. ARCADE GAME.....	7
3.1.1. <i>Java PacMan</i>	8
3.1.1.1. Hardware and Software Requirements.....	9
3.1.1.2. Setup and Run.....	9
3.1.1.3. Usage	9
3.1.2. <i>Web PacMan</i>	9
3.1.2.1. Hardware and Software Requirements.....	10
3.1.2.2. Setup and Run.....	11
3.1.2.3. Usage	11
3.2. SOCIAL GAME.....	12
3.2.1. <i>Hardware and Software Requirements</i>	13
3.2.2. <i>Setup and run</i>	14
3.2.3. <i>Usage</i>	16
3.2.4. <i>Mobile GUI design</i>	28
4. CONCLUSIONS.....	31

1.Introduction

The aim of this document is to provide instructions on how to install, run, and use the application prototypes for the Business Domain and the Game Domain, respectively. The document is organized into three sections: section 1 is a general introduction; section 2 and section 3 present the hardware and software requirements, the installation instructions and the usage of the prototypes that have been developed for the two domains, business and game.

2. Business Domain Prototype

This section describes an initial application prototype for the business domain, together with the instructions on how to install, run, and use it. Currently, this is a rudimentary, yet foundational, application enabling experimentation. Over time it will be elaborated to encompass more powerful functionality. Moreover a first discussion of the challenges which the business applications raises in regard to the design of a language to logically describe UIs and the use of that language to support UI migration has been presented in D2.3.

In order to deliver desktop-like behavior and interactivity, the application is implemented as a Rich Internet Application (RIA). The implementation of this RIA utilizes Microsoft Silverlight (<http://silverlight.net/>), which is a cross-platform web application framework for RIAs based on the .NET technology and a web-based subset of the Windows Presentation Foundation (WPF). Silverlight offers a wide range of animation and multimedia capabilities as well as lean connectivity to the backend.

Silverlight applications are developed using a declarative user interface description language called XAML (eXtensible Application Markup Language). XAML is used as a user interface markup language to define UI elements, data binding, eventing and other features. Silverlight applications are developed using XAML, and either JavaScript, Visual Basic, C# or Ruby for coding. The initial business application is implemented in C#, and has been developed using Microsoft Expression Blend 2 for the UI design and Microsoft Visual Web Developer 2008 Express Edition for the backend coding. This application is map-based, using Microsoft Virtual Earth (maps.live.com) to display and control maps. In order to use the benefits of managed code for client-side scripting, the initial business application utilizes VIEWS (Virtual Earth Wrapper for Silverlight) (<http://www.codeplex.com/views>), a wrapper around the JavaScript Virtual Earth control.

2.1. Hardware and Software Requirements

The business application requires a browser and an Internet connection. The Internet connection is mainly needed to access maps provided by Microsoft Virtual Earth.

Silverlight supports most platforms and browsers: Internet Explorer and Mozilla Firefox are supported on the Windows platform, Mozilla Firefox and Apple Safari are supported on the Apple platform.

There are no particular hardware requirements. A standard reasonably powerful desktop PC is sufficient.

2.2. Setup and Run

The business application runs in a browser and requires that the Silverlight Plugin downloaded and installed. This occurs by means of a One Click Install from Microsoft the first time the Silverlight business application is called. Figure 1 depicts the application in the browser and indicates how a backend system can be added to future versions of the application to provide additional functionality.

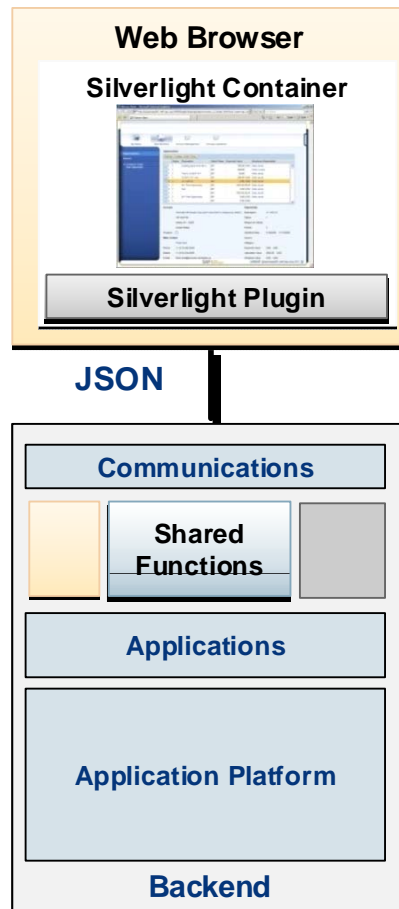


Figure 1: Overview of the Architecture of a Silverlight Application

2.3. Usage

This first application is relatively simple, and is principally intended to demonstrate the usefulness of migration as well as some ideas about how to combine multiple applications on a smartwall, which is a device with a large high-resolution display.

The scenario is that two experts work on different aspects of the same problem and then merge their results onto the smartwall. In the foreseen scenario, two experts are involved: a flooding expert and an evacuation expert. Both simulate their respective scenarios, thus, creating animations of flooding and evacuation, respectively. These animations are geo-referenced time-sequenced datasets that are overlaid on the referenced map at the referenced time, thus creating the effect of an animation on top of the map.

To show migration of the applications, first, one simulation is started and migrated to a free smartwall. Next, the second simulation is started and migrated, but only the presentation elements are involved in the migration. The control elements are merged with those of the first simulation. In this way, the simulations are synchronized, e.g. same starting and end times, same rate of running animations,

etc. Note that the parameter specifying the initial datasets used for the simulations are not merged, since the whole point is to compare the results of different input datasets.

The UIs of the applications for this first version are pretty much the same. The main difference is which animation is picked for display on the desktop. Aside from the map display with its overlaid animation, there are two main sets of controls: one for the animation and one for the map. An overview of the first draft of the UI, which is still subject to change, is shown in Figure 2.

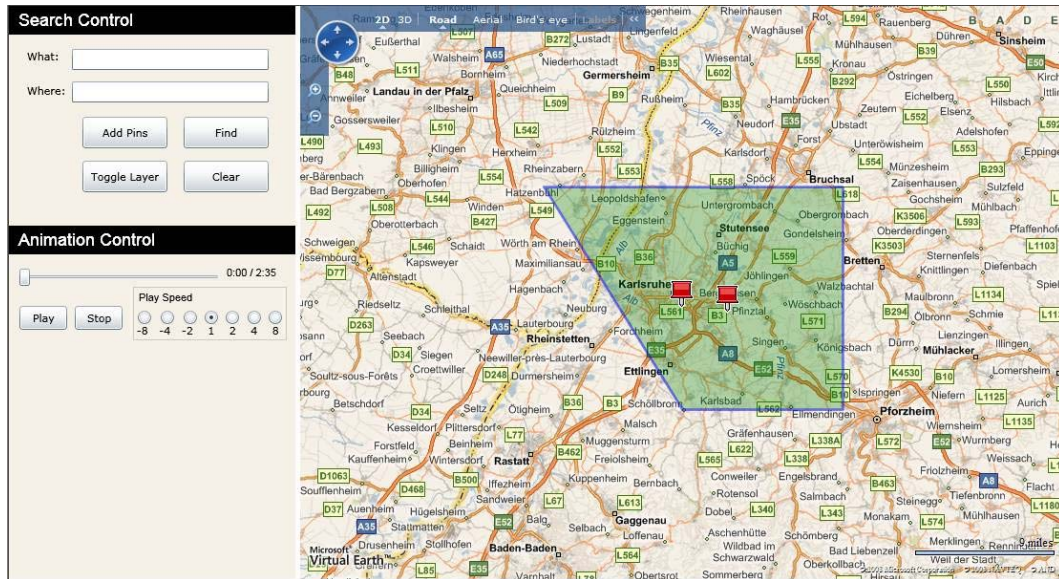


Figure 2: First Rudimentary UI

The set of animation controls are shown in Figure 3. They consist of a slider for moving the time, a speed control and the buttons for starting (or resuming) and stopping the animation.



Figure 3: Animation Controls

The map controls, shown in Figure 4, are a standard part of the functionality that comes with Microsoft Virtual Earth. There is a compass that can be used to pan in any direction, a zoom-in button and a zoom-out button as well as the capability to change the center point of the map, and to move it by clicking and dragging. For

more information see maps.live.com. Other controls shown in Figure 4 are not actually used in the application and will eventually be blended out.



Figure 4: Virtual Earth Map Controls

3. Game Domain Prototypes

This section describes the application prototypes for the Game domain, together with the instructions on how to install, run, and use them.

3.1. Arcade Game

An arcade game was formally a coin-operated entertainment machine, typically installed in businesses such as restaurants, pubs, video arcades, and Family Entertainment Centres. Arcade games are often characterized by very short levels, simple and intuitive control schemes, and rapidly increasing difficulty. This is due to the environment of the Arcade, where the player is essentially renting the game for as long as her/his in-game avatar can stay alive or until she/he runs out of tokens. Nowadays, many arcade games have been ported to the PC and enjoy great popularity as they can be played not only on PCs, but also on resource limited devices like mobile phones or PDAs, for example.

PacMan is a classic arcade game where a figure called PacMan, which is steered by the user, has to collect dots in a maze, as shown in Figure 5 **Errore. L'origine riferimento non è stata trovata.**

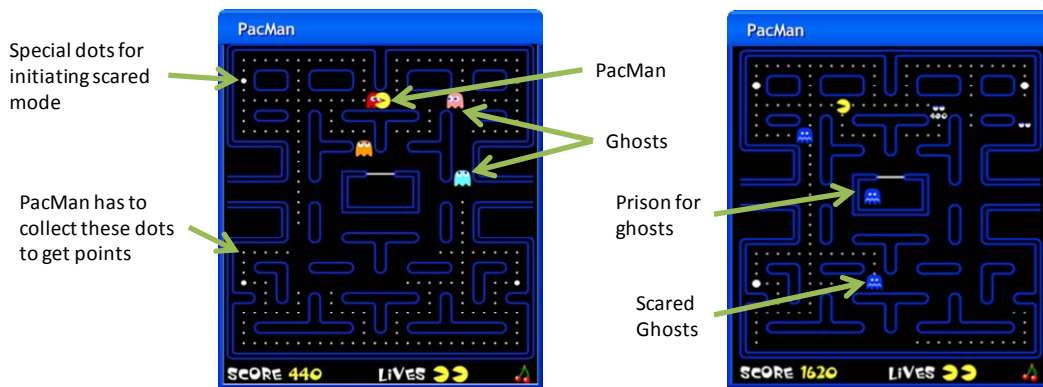


Figure 5: The two modes of a single PacMan game. On the left hand side the game is in normal mode where the ghosts try to catch the PacMan. On the right side the game is in scared mode where the PacMan can catch the ghosts.

Ghosts, who are controlled by the computer, are running around, with the goal to catch the PacMan. If the PacMan collects special dots, ghosts and PacMan change roles for some seconds, as depicted in **Errore. L'origine riferimento non è stata trovata.** on the right hand side. It means that the PacMan now can catch ghosts while the ghosts try to run away. Caught ghosts will be imprisoned in the middle of the maze for some seconds; afterwards, the roles change back again. The goal for the player is to get as much scores as possible by collecting dots and catching ghosts.

Because of its simple application logic and slim implementation, the PacMan game lends itself as an interesting prototype for presenting migration and adaptation.

In order to cover a wide range of technologies, as preliminary experiments for the prototypes, two versions of PacMan implemented in different languages have been taken into consideration, namely Java PacMan and Web PacMan.

3.1.1. Java PacMan

To illustrate migration and adaptation aspects from a logic reconfiguration point of view, two main enhancements to the standard PacMan game have been developed. In the first scenario, a single PacMan game is migrated from a PC to a PDA and vice versa, as depicted in Figure 6.



Figure 6: Migration of a single Pacman game from a PC to a PDA

For example, if the user is playing on his PC and she/he has to leave, she/he can migrate the game to her/his PDA and play during the bus trip. The user will then see that the game adapts to the target platform. The number of dots within the maze may change due to the smaller size of the PDA screen. From this follows that the scoring has to be adapted in a way that to collect one dot on a PDA results in higher scoring. Other adaptations are that the speed of the ghosts may slow down since it is more complicated to steer the PacMan on a PDA.

In the second scenario, multiplayer functionality to the PacMan game has been added. If a second player also starts a PacMan game on his PDA, he can join the other game. The result is that a caught ghost is not imprisoned in his own maze, but it will appear in the game of the opponent player as shown in Figure 7.

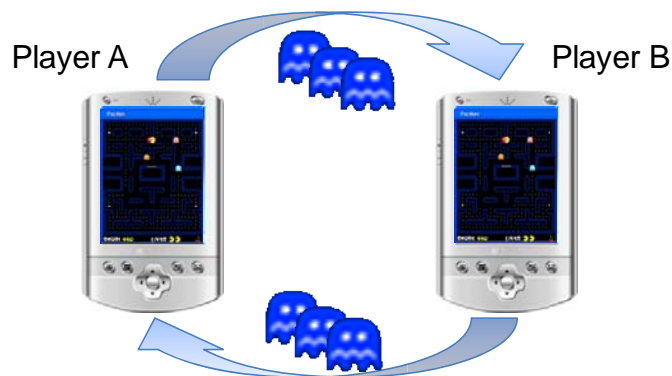


Figure 7: Two players play against each other by sending ghosts to each other.

In this way, the players play against each other by sending ghosts to the opponent in order to make the game more complicated for the opponent.

3.1.1.1. Hardware and Software Requirements

The single Pacman game can be run on a PC, as well as on PDAs or mobile phones provided that the device has a Java virtual machine running on it. Anyway, there are various JVMs available for nearly all kinds of devices. As we use Java, the game can be executed on any operating system. In single player mode, the game does not need a network connection. However, to realize migration functionality, at least a connection between all the devices is required. Furthermore, depending on the realization of the migration functionalities, a connection to a migration server may be required. Finally, on each device, a client-version of the OPEN platform has to be executed, which is among others responsible for device discovery, context gathering, migration, and adaptation of the user interface and application logic.

3.1.1.2. Setup and Run

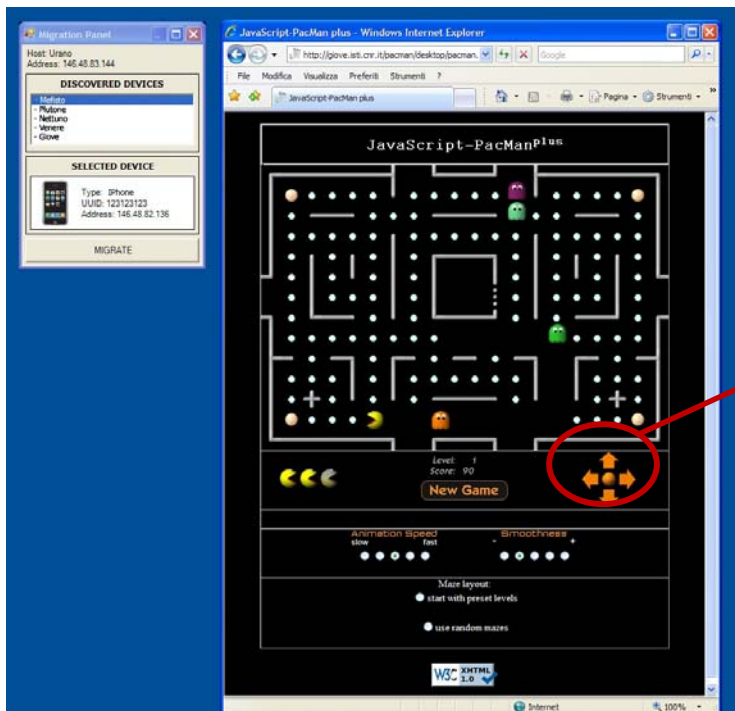
Depending on the realization of migration and adaptation within the middleware, it has to be ensured that an OPEN server is running. Furthermore, the user has to start the OPEN client which could be delivered as an executable JAR file. The game itself could also be delivered as an executable JAR file which the user can simply start by clicking on an icon.

3.1.1.3. Usage

On the PC the user can use the arrow keys on the keyboard to steer the PacMan. Migration could be initiated by using a specific menu in the OPEN client. There she/he can choose between available target devices for migration. After the migration, the player can resume the game by using the OPEN client menu.

3.1.2. Web PacMan

In this section we will describe the current implementation of a migration applied to the Web PacMan game, which has been carried out by migrating from a desktop platform to a mobile one. In Figure 8 and Figure 9 are shown the desktop version and the mobile version respectively.



Imagemap: arrow-shaped buttons for steering the pacman, plus a central round-shape button for pausing the game

Figure 8: Migration Client (top-left part); Desktop version of the PacMan example (right part)

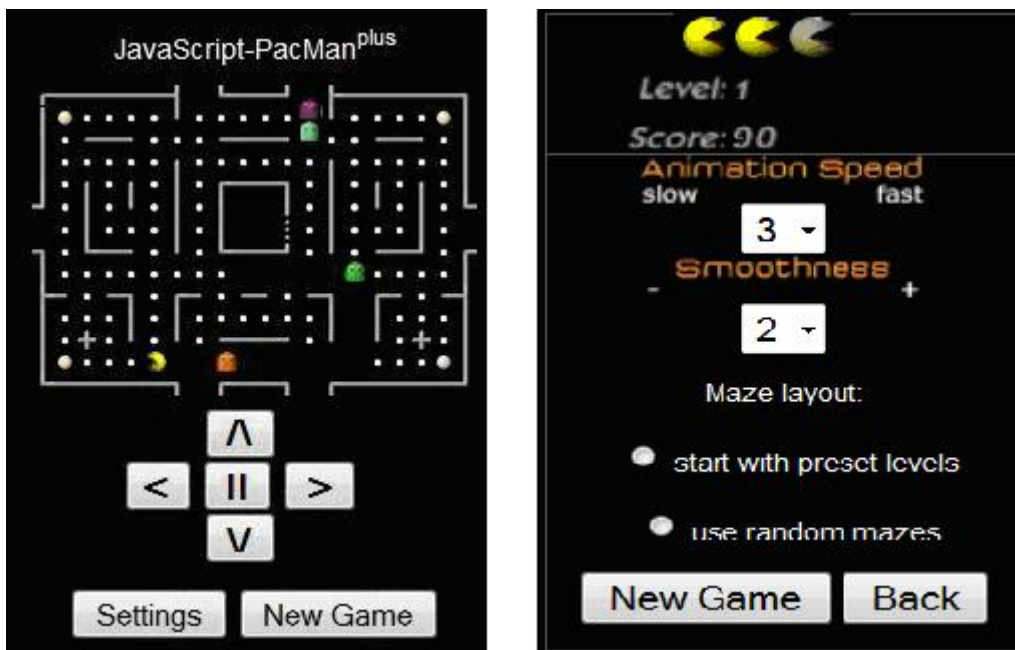


Figure 9: The two pages of the mobile version of the PacMan example generated after migration

3.1.2.1. Hardware and Software Requirements

The prototype currently implemented at ISTI-CNR is developed using Java version 5 (or higher), and it is based on a Migration Server which is aimed at coordinating the various phases of the migration, which uses Apache Tomcat 5.5.

Since the Migration Server has to be aware of the available devices in the environment, it uses a Device Discovery module which is based on multicast

datagrams using UDP/IP. In order to activate the migration towards a specific device, on the source platform a Migration Client has to be available, allowing to select the migration target among the devices currently available for migration. The Migration Client is a tiny program developed in C#, and, in order to work properly, it needs .NET Framework on desktop platforms and .NET Compact Framework on mobile devices.

For the PacMan example we have used an implementation of the game for the desktop platform in which both XHTML 1.0 (presentation part) and JavaScript (mainly devoted to the application logic part) are used. In order to support a migration from desktop to mobile device, the desktop version of the PacMan game has been transformed in order to obtain an adapted version for the mobile device, implemented in XHTML Mobile Profile language. Therefore, on a desktop platform, the PacMan example can be executed by a browser that can interpret both XHTML and JavaScript (nowadays practically all the browsers for desktop platforms can interpret such languages, provided that JavaScript support is activated). Regarding the mobile devices, browsers like Safari (e.g. on the iPhone) and Opera have been verified to work well for the PacMan application example in which also JavaScript excerpts are included.

3.1.2.2. Setup and Run

In order to support migration, it has to be ensured that the Migration Server is running and properly working. In addition, the devices have to be able to communicate with the Migration Server, which will coordinate the various phases of the migration process. Therefore, in order to do this, before starting, all the client platforms that want to subscribe to the migration service should run the Migration Client.

3.1.2.3. Usage

The scenario is of a user that starts to interact with the desktop version of the PacMan (which has been previously downloaded and annotated with Ajax scripts by the Migration Server, in order to be able to save state information). On the desktop platform the user can use both a keyboard-based interaction (by using arrow keys or other predefined keys) and mouse-based interaction (e.g. by interacting with the imagemap displayed under the maze, see Figure 8 – right part). Indeed, in the one-page desktop version of the game different components can be found:

- the maze of the game, in which the ghosts and the PacMan will be visualized during the game;
- a part mainly devoted to the visualization of the current state of the game (e.g., the number of PacMan lives still available, the level of the game and the score, ..);
- interaction elements for controlling the game: the 4 arrow -imagemap, used for selecting the direction of the PacMan (the user can either simply put the mouse over the concerned region of the imagemap for selecting the corresponding direction, or explicitly click on the concerned region); two controls for starting a new game (implemented with the central ‘New Game’ button in Figure 8), and pausing the game (implemented with the

central round-shape button of the imagemap). Also, as previously mentioned, in the desktop web page, it is also possible to use the keyboard (e.g.: by using the arrow keys or other predefined keys –'j', 'i', 'l', 'k', for the steering the PacMan directions, and 'p' and 'n' keys, to pause and start a new game respectively).

- additional settings for specifying the current configuration of the game (e.g.: the animation speed of the PacMan, the possibility of using random maze layouts for the levels, etc.).

After playing for a while the game on the desktop platform, at a certain point the user decides to migrate towards a mobile device. Therefore, the user has to select, by interacting with the Migration Client (see Figure 8, top-left part), the platform towards which the migration will be activated, among the available devices. After doing this, the application will be migrated on the new platform.

When migrating to the mobile device, the original page will result to be split into two pages (see Figure 9) as a result of an adaptation phase, since the more limited capabilities of the target mobile device do not allow to include all the elements in only one page.

The adapted version for the mobile version is displayed in Figure 9. As you can see, two pages have been generated as a result of adaptation. In the first page the user can control the game through five buttons (the meaning of the labels is: '<' = left, '>' = right, '^' = up, 'v' = down, '|'| = pause). Moreover, the first page of the version adapted for the mobile device contains the button allowing the user to start a new game. An additional page (which is reachable by interacting with the "Settings" button on the first page) will include all the remaining elements (current PacMan lives, current level and score, settings for animation speed and smoothness). Moreover, also other controls are redesigned in order to be adapted for the mobile device (see Figure 9, right part): for instance, radio buttons with several choices will be replaced by pull-down menus, in order to save space on the target (mobile) device.

Regarding the state of the game, it should be preserved and then re-activated in the mobile device at the point where the migration was activated. Therefore, the current PacMan and ghost positions should be saved after migrating onto the mobile platform, together with the current level and score and all the other settings that the player already selected in the desktop version, in order to allow the user to continue the interaction with the application from the point where s/he left off.

3.2. Social Game

The Social Game is a complex web application organized into several elements: Chat, Betting, IPTV, Additional Content, Racing Game, which have been described in details in D5.1 - Initial application requirements and design. In particular, the Racing Game is a client-server application, where the client is hosted inside the web page as an embedded object through a native plugin that

needs a specific setup and configuration to work properly, and will be specifically described in the next paragraphs.

In order to simulate the migration of the game controls on a remote device, an additional game client for mobile phones is also available, whose setup and usage will be described in specific paragraphs.

Together with the prototype instructions, in this section a first design of how the GUI of the web application should look like on the mobile phone is presented.

3.2.1. Hardware and Software Requirements

To properly run the Social Game prototype, a specific hardware and software configuration is needed.

The Racing Game is implemented as set of three applications: a client running inside the browser and two separate servers, one devoted to handle physics computations, the other one to manage the game logic. In principle, each of them can be run in a separate host, with the following minimum requirements:

- **Game Client:** Intel or AMD CPU running at 2Ghz with 1Gb RAM, ATI or NVIDIA GPU supporting Shader Model 2.0, and 100Mb of free disk space, Display with 1280x1024 resolution; Mozilla Firefox 3.x with JavaScript enabled running on Windows XP SP2 operating system.
- **Game Logic Server:** Intel or AMD CPU running at 1 Ghz, 256 Mb RAM, 100 Mb of free disk space; Windows XP SP2 or Linux operating system.
- **Physics Server:** Intel or AMD CPU running at 2 Ghz, 512 Mb RAM, 50 Mb of free disk space; Windows XP SP2 or Linux operating system.

Regarding the network configuration, each application needs a network connection with at least 2 Mbit bandwidth and a set of open TCP and UDP ports (see below). Moreover, an internet connection is needed.

Anyway, all the applications can be run on a unique host, provided that a more powerful hardware configuration is used. In this case, the minimum requirements are the following:

- Intel or AMD Dual Core CPU running at 2.2 Ghz, 1 Gb RAM, ATI or NVIDIA GPU supporting Shader Model 2.0, network adapter with an internet connection, and 100Mb free disk space; Windows XP SP2 operating system.

For the additional mobile game client the following requirements are needed:

- A mobile phone supporting Connected Limited Device Configuration (CLDC) 1.1 and Mobile Information Device Profile (MIDP) 2.0 (e.g. Symbian OS based Nokia Series 60 Phones).
- UMTS, H3GA or wifi network connection
- Permission for network access

This version has been tested on Nokia N95 and Nokia N93 phones. Other phones may be compatible with this release, but they were not tested.

3.2.2. Setup and run

To install all the applications under Windows operating system, a self-extracting setup package *setup_opengame.exe* has been provided, which allows selecting the applications to be installed. The typical configuration consists of two servers running on one PC and a number of clients running in different machines, connected to the servers through a LAN or an Internet connection. As a particular case, one client and the servers can be run on the same machine.

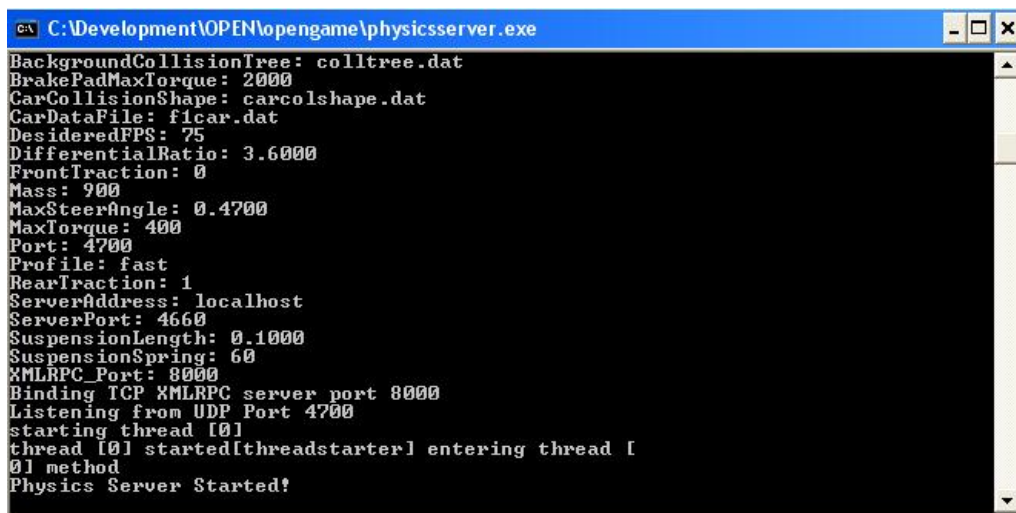
During the installation process, please do not change the proposed destination folder, to allow the correct operation of the software.

In all the PCs that run the Social Game, the openGame plugin for Firefox must be installed. To this aim, the *open_plugin.xpi* package is provided. You can simply drag the file into the Firefox window and follow the instructions.

In order to support the gaming environment, both the Game and the Physics Servers must be run and configured for listening to clients connections.

First of all, launch the Physics Server through the corresponding shortcut in the start menu. By default, TCP port 4700 and UDP port 8000 are used for communicating with the Game Server. In case you need to change these ports, you can edit the “Port” and “XMLRPC_Port” settings in the configuration file *physicsserver.cfg*, which is located in the installation folder.

After launching the Physics Server, wait for the message “Physics Server started!” in the open console, as shown in Figure 10.



```

C:\Development\OPEN\opengame\physicsserver.exe
BackgroundCollisionTree: colltree.dat
BrakePadMaxTorque: 2000
CarCollisionShape: carcolshape.dat
CarDataFile: ficar.dat
DesideredFPS: 75
DifferentialRatio: 3.6000
FrontTraction: 0
Mass: 900
MaxSteerAngle: 0.4700
MaxTorque: 400
Port: 4700
Profile: fast
RearTraction: 1
ServerAddress: localhost
ServerPort: 4660
SuspensionLength: 0.1000
SuspensionSpring: 60
XMLRPC_Port: 8000
Binding TCP XMLRPC server port 8000
Listening from UDP Port 4700
starting thread [0]
thread [0] started[threadstarter] entering thread [
0] method
Physics Server Started!

```

Figure 10: Physics Server console window

Then, launch the Game Server through the corresponding shortcut in the start menu. By default, UDP port 4660 is used to communicate with the Physics Server and the clients. If you need to change this port, you can edit the “Port” setting in the configuration file *server.cfg*, located in the installation folder. Note that, in this case, you need to change also the “ServerPort” setting in the Physics Server configuration file *physicsserver.cfg*.

Once both servers are up and running, you can open the Social Game web page through the corresponding shortcut in the start menu. Before that, you should

check if the client is properly configured to connect to the servers. To do this, open the client configuration file *client.cfg* and modify “ServerAddress” and “ServerPort” settings with the IP address and UDP port where the Game Server is hosted. If the client and the servers are running on the same machine, you should use “localhost” or “127.0.0.1” as “ServerAddress”.

Note that clients by default use UDP port 4662 for incoming connections. You can change this by modifying the “Port” setting in *client.cfg*.

Finally, each player should choose a different name by editing the “Name” setting in the client configuration file *client.cfg*.

To install the OpenGame mobile client midlet you need to copy the *OpenGame.jar* file to your mobile phone. Different phones have different procedures to install new midlets. The procedure presented in this document shows the process used to install the midlet onto a Nokia N95 mobile phone using a computer running Windows XP SP2.

The mobile phone needs to be switched on and Bluetooth must be enabled. For detailed instructions about how to activate Bluetooth on your phone, please follow the mobile phone manual instructions. The Bluetooth protocol also needs to be supported and enabled in your computer. Please refer to your computer manufacturer manual for how to enable Bluetooth. Different combinations of computers and mobile phones can use different connections to transfer files, like USB, Serial or Infrared. The following procedure only covers the file transfer using the Bluetooth protocol:

- Check if the Bluetooth connection is set to On your mobile phone and if the phone's visibility is set to “Shown to all”.
- Locate on your PC the file *OpenGame.jar* and transfer it through Bluetooth to your mobile phone.
- Once the file has been transferred, go to the messages folder and look for the *OpenGame.jar* message.
- Open the message and follow the instructions (Note that the midlet is untrusted, install it anyway)
- From the Main Menu open the “Applications” folder, you should see a screen like that shown in Figure 11
- Select the OpenGame Midlet icon and launch it

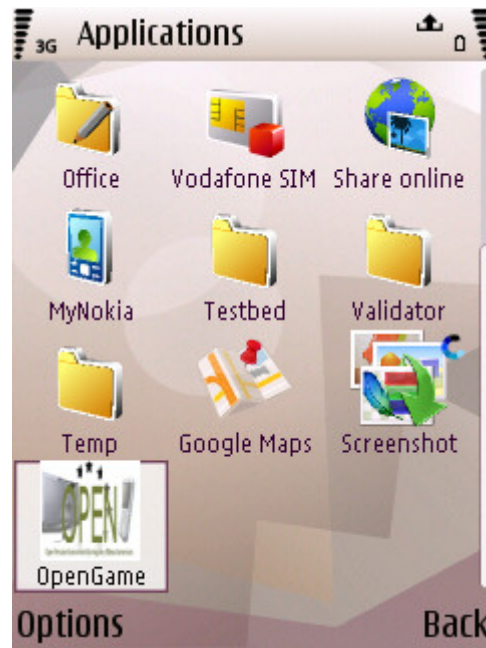


Figure 11: Nokia N95 Applications Menu

3.2.3. Usage

This section explains how to use the Social Game prototype that has been described in D5.1.

Once the web page of the Social Game is shown into the browser, press F11 to switch to full screen mode. The page will look like Figure 12. Since you are not yet logged, only the IPTV and additional info are active, whereas the Racing Game (greyed) and the chat and betting elements are not available.



Figure 12: Social Game page before logging

The IPTV is simulated on the top left of the screen by a Flash video plugin that shows a Grand Prix video. In Figure 13, an enlargement of the IPTV element is shown. It is possible to switch among the channels, pause and restart the video, as well as controlling the volume and switching to full screen mode, through suitable buttons.



Figure 13: IPTV enlargement

On the right of the IPTV some info about the race are displayed (see Figure 14).

Gp Valencia



Valencia is the capital of the Spanish autonomous community of Valencia and its province. It is the third largest city in Spain and the 21st largest in the European Union. It forms part of an industrial area on the Costa del Azahar. The estimated population of the city of Valencia proper was 797,654 [1] as of 2007 official statistics. Population of the metropolitan area (urban area plus satellite towns) was 1,738,690 as of 2007.

Game Position		Live Position
Pos	Num	Driver
01	03	Danny The Dog
02	08	Felipe Massa
03	02	Kimi Raikkonen
04	12	Felix TheCat
05	06	Mark Webber

Figure 14: Additional Info enlargement

In particular, you can see alternatively the live race positions or the live race positions merged with the Racing Game positions, by selecting the corresponding tab. Note that the shown information is simulated and does not depend on real data.

To be able to use all the elements of the Social Game, you need to insert your login and password in the fields on the top right of the page. After entering your

authentication data, you will have access to the remaining elements, i.e. the Chat, the Racing Game and the Betting, as shown in Figure 15.



Figure 15: Social Game after authentication

The chat area is on the right of the additional info. An enlargement is shown in Figure 16.

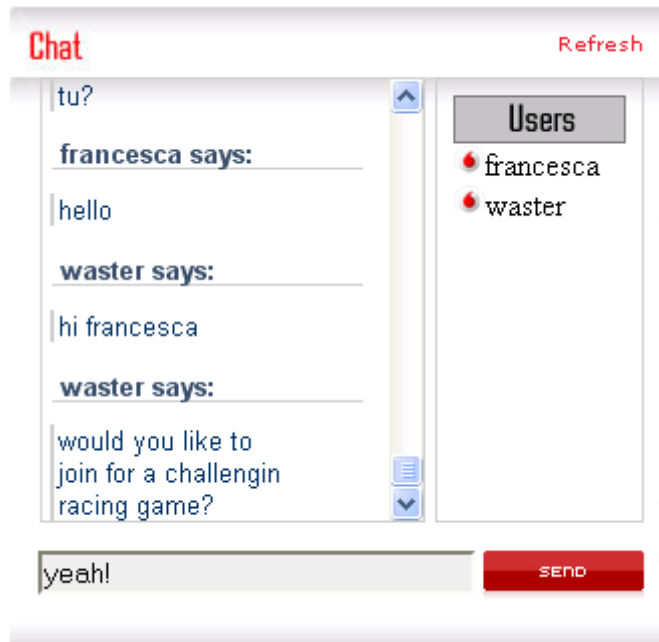


Figure 16: Chat area enlargement

You can see the buddy list on the right and the conversation log on the left. In case of network problems, the Refresh button will force reloading the conversation history.

In Figure 17 an enlargement of the betting area is shown. On the top, tipsters about pilots are listed. Below them, bet results are shown. You can select which parameter and driver you want to bet on, as well as the desired amount.

BETTING AREA

Tipster

Name	Quote
Lewis Hamilton	4/6
Felipe Massa	11/5
Kimi Raikkonen	9/2
Robert Kubica	80/1
Mark Webber	80/1

Bet Results

Valencia 2008 - winner
Lewis Hamilton - 26

Bet Parameters

Championship

Best Lap
 Pole Position
 Winner
 Fastest Lap

Driver

Kimi Raikkonen (FIN/Ferrari) ▼

Bet Import

, €
BET

Figure 17: Betting area enlargement

Pressing the BET button, the selected betting parameters are shown and you are asked to confirm your choices, as shown in Figure 18.

Your Bet Details

Username: john

Import: 125,00 €

Bet Parameters: Winner

Driver: Felipe Massa (BRA/Ferrari)

SUBMIT

CLOSE X

Figure 18: Betting details

Pressing the SUBMIT button, you can choose between two payment options: mobile accrediting and credit card, as shown in Figure 19.

Your Bet Details **Payment** **Review and Confirm**

Payment

Name: John
Surname: Smith

Select Payment Mode:

Accrediting Mobile **Credit Card**

CLOSE X

Figure 19: Betting options

If you choose Accrediting Mobile, you can select or confirm your mobile number (see Figure 20).

Your Bet Details **Payment** **Review and Confirm**

Payment - Accrediting Mobile

Name: John
Surname: Smith
Mobile Number: +39 3493678607

SUBMIT

CLOSE X

Figure 20: Betting user details

To submit your bet, press the SUBMIT button. If your request is accepted, you will see a confirmation message (see Figure 21).

Your Bet Details **Payment** **Review and Confirm**

Review And Confirm

Name: John	Username: john
Surname: Smith	Import: 125,00 €
Mobile Number: +39 3493678607	Bet Parameters: Winner
	Driver: Felipe Massa (BRA/Ferrari)

CONFIRM

CLOSE X

Figure 21: Betting resume

In the bottom right of the Social Game web page, the Racing Game area is displayed. The aim of the game is to perform a lap in the shortest time. As soon as you logged, a car is assigned to you. In order to be able to drive your car you must

click on the Racing Game area. Use the arrow keys to control the car: up arrow for accelerating, down arrow for braking, left arrow to steer left and right arrow to steer right. Although the gearbox is automatic, keypad plus and minus allows increasing and decreasing car gears sequentially. Decreasing the first gear will set the neutral gear; decreasing the neutral gear will set the reverse gear.

Keys from 1 to 3 allow changing the camera point of view: free, external and internal, respectively. To control the free camera, use WSAD and RF keys. To restart your race and start from the pitlane, press the T key.

In Figure 22 a screenshot of the Racing Game is shown.



Figure 22: Racing Game area enlargement

On the top right of the screen the player name is displayed in red, together with his current and best lap times. Below them, the name and the time of the player who performed the best lap is shown.

On the bottom right of the screen three indicators are drawn in foreground: the speed, the gear and the engine RPM.

The Mobile Client is a J2ME client application that allows the user to control her/his car with his mobile phone while playing on the PC, as well as showing some relevant information on the mobile phone display.

In Figure 23 a portion of Nokia N95 keyboard is shown. The Up, Down, Left and Right keys will be called Navigation Keys, and the Right and Left options keys will be called Right and Left Softkeys.

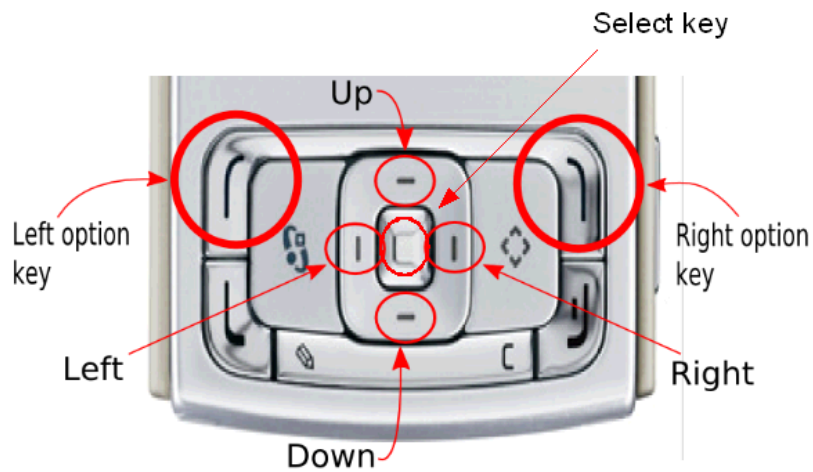


Figure 23: Nokia N95 keyboard



Figure 24: OpenGame midlet Main Menu

You can use the Up and Down Keys to navigate the menu and the Select key to select a menu entry.

Navigate the Menu and select the Settings entry, shown in Figure 25.

Settings

GAME SERVER
192.168.1.76:4660
PLAYER NAME
Francesca
STEER SENSITIVITY
2
ACCELERATOR SENSITIVITY
1
BRAKE SENSITIVITY
1

Ok

Back

Figure 25: OpenGame midlet Settings Menu

The Settings Menu allows to change the address of the Game Server, the player name, and to set some sensitivity parameters for controlling the car.

To navigate the Settings Menu use the Up and Down Keys. In order to allow the connection to the Game Server, you must insert the same address, the port and the same player name you entered in the *client.cfg* file.

The sensitivity parameters allow changing the control sensitivity of the corresponding control. For example, a high steer sensitivity provides a faster rotation but a poor accuracy. To change a sensitivity, navigate to the desired field and use Left and Right keys to decrease or increase the sensitivity parameter.

To save the Settings, select Ok by pressing the Left softkey: to discard the changes, select Back with the Right softkey. In any case you will be brought back to the Main Menu.

For start playing, select Play and you will be asked to be connected to the Game Server, as shown in Figure 26.

Connecting



Figure 26: Network permission request

The game client demands your permission to use the network; to proceed select Yes, and then select your Access Point for network connection (see Figure 27).



Figure 27: List of access points

Once the client is connected to the Game Server, you will see the screen in Figure 28.



Figure 28: Game Screen

The GUI shows the Current Lap Time of the player, her/his Best Lap Time and the Record Time together with the name of the player who performed it. The car can be controlled using UP key to accelerate, DOWN key to brake and LEFT and RIGHT keys to steer. To disconnect from the Game Server, select Exit option with the Right Softkey.

3.2.4. Mobile GUI design

As explained in D5.1, part of the Social Game web page could be migrated to a mobile device. In order to study a possible layout of the migrable elements, a first partial design of how the mobile GUI of the web application should look like is presented in this paragraph. It is worth noting that this design is not meant to be the final one, but only a suggestion for future development.

As mentioned in D5.1, because of the limited performances of mobile devices, the IPTV will not be migrated. Thus, only the controls and part of the GUI of the Racing Game have been considered for the migration. In order to cope with the reduced screen size and the different input interface, the web page has been split in several parts, which can be navigated through tabs.

In Figure 29 three screenshots of the Additional Info element are shown. The first one shows general information about the live race, the second one the live race positions, and the third one the live race positions merged with the Racing Game positions.



Figure 29: Additional Info tabs

In Figure 30 the Betting tab screenshots are shown, together with the tipsters, the bet details and the payment options.

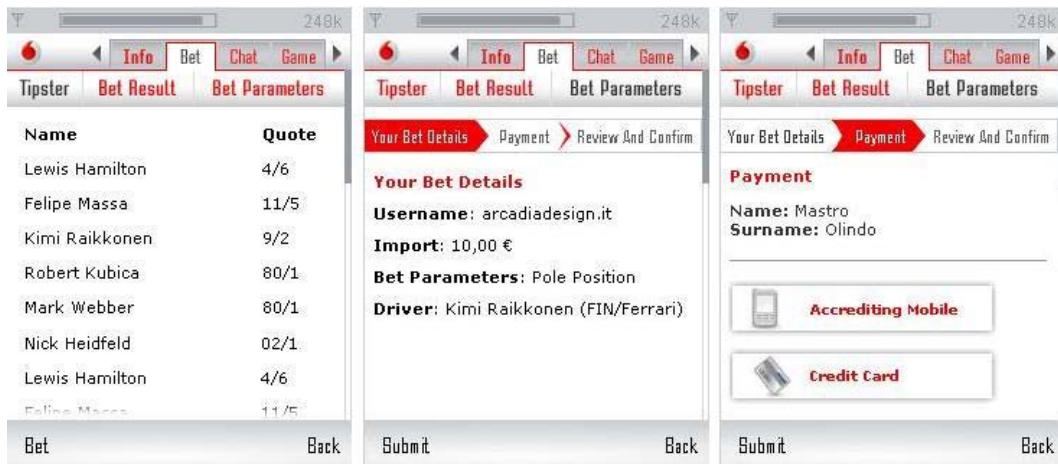


Figure 30: Betting tabs

In Figure 31 and Figure 32 the two payment modalities, respectively credit card and accrediting mobile are displayed.



Figure 31: Mobile payment



Figure 32: Credit card payment

In Figure 33 a possible screenshot of the dialog screen of the chat is shown. Moreover two solutions for the GUI of the game are suggested.



Figure 33: Chat and Racing Game screens

4. Conclusions

In this document a description of the software prototypes developed for both the Business and Game domains has been presented. These prototypes should be considered as a starting point for experimenting with the migration platform.

Regarding the Business domain, the scenario initially implemented serves to demonstrate the usefulness of migration as well as to explore the implications of migration of multiple applications to the same device. Future versions will extend the functionality of the applications to explore the research challenges in more depth.

As for the Game domain, the Web PacMan has been used for experimenting on UI adaptation and migration from desktop to mobile device, whereas the Java PacMan for Logic Reconfiguration from single player to multiplayer. To better analyze the implications of the migration aspects in an application involving interactions among different components, a first version of the Social Game prototype has been implemented.