Simon Says the Color: The Digital Evolution of an Outdoor Kids Game

Claudio E. Palazzi
Dipartimento di Matematica Pura e Applicata
Università di Padova
via Trieste, 63, 35121 Padova, Italy
cpalazzi@math.unipd.it

Dario Maggiorini
Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano
via Comelico, 39, 20135 Milano, Italy
dario@dico.unimi.it

Andrea Burattin
Dipartimento di Matematica Pura e Applicata
Università di Padova
via Trieste, 63, 35121 Padova, Italy
burattin@math.unipd.it

Riccardo Ferro
Dipartimento di Matematica Pura e Applicata
Università di Padova
via Trieste, 63, 35121 Padova, Italy
rferro@studenti.math.unipd.it

ABSTRACT

In the last decades mobile phones have evolved from simple instruments to make calls to complex and powerful devices. Following this trend, a popular application as gaming has extended its domain from home computers and consoles to the mobile realm. Mobile games represent now a killer application for smart phones and are attracting millions of subscribers worldwide. Nowadays, the challenge is that of creating a new generation of mobile games proposing something more than just a tiny version of home entertainment blockbusters; the ubiquitous use and the interesting features (camera, gyroscope, connectivity, GPS, etc.) of mobile phones should be fully exploited to generate a completely new kind of gaming experience. To this aim, we have created an application for mobile phones exploiting new features available in every modern mobile phone (in particular, bluetooth communication and camera), to digitally replicate and enhance an outdoor game for children that would not be playable (with fun) on an desktop PC.

Categories and Subject Descriptors

K.8.0 [**Personal Computing**]: General—*games*

General Terms

Design, Experimentation

Keywords

Mobile Games, Mobile Devices, Games, Wireless

1. INTRODUCTION

We are living in a revolutionary era for communication: the number of mobile phones have surpassed landline ones in many countries (such as Italy and Finland). These devices, in the last few years, have increased their capabilities in terms of memory storage, CPU power, and connectivity (e.g., GPRS, UMTS, Wi-Fi, Bluetooth). Furthermore, the use of well known and open software platforms (like Symbian [1] and Android [2]), and the free availability of development frameworks (like Java ME) has attracted the business world together with researchers from various fields: gaming, networking, artificial intelligence, communication, imaging, and many others. As a result from this growing interest the mobile customers evolved from simple callers to messengers (first with SMS, then using MMS) and now they are becoming users of complex applications [3], [4]. Great examples of applications can be found on the Apple iPhone online applications store [5], which has been the first large-scale mobile applications distributor. In particular, market analysts report incomes for the wireless gaming industry in the order of billions of US dollars worldwide and with a 40% growth each year [6].

Gaming industry is moving from the classical, first generation, way to play (closed in a room, using a joypad) to new experiences, particularly focusing on two fronts:

1. allowing to play everywhere, with always better visual effects (e.g., Sony Play Station Portable) and
2. using controllers that require a physical control on the game (e.g., the Wii Remote and the Wii Balance Board, for the Nintendo Wii console).

Our approach tries to merge the above points: an ubiquitous game combined with strong physical involvement. We believe this purpose can be reached with common (and low priced) devices, such as mobile/smart phones, using the typical equipment supplied with them, such as bluetooth connection and a coarse camera.

Classic games are simple to be played and they do not require high performing hardware; it is hence relatively easy to develop these games for modern mobile phones that are computationally equivalent to old computers. Old amusement arcade games have wide diffusion between mobile applications, but they fall short to exploit the potential of new technologies [7], [8]. New technologies permit to make old games more amusing and to increase their fun potential through remote gaming, impartial automatic arbitrage, score ranking, augmented reality, etc. The small size of mobile phones and easy games allow people to have fun everywhere and anytime.

We created an application that digitally mimics an old outdoor

Italian game for kids named "*Strega Comanda Colore*" (*Color Commanding Witch*) which, in the english culture, may be more familiar as "*Simon Says*" with the goal to seek out colors. For sake of simplicity, from here on we call this game and the corresponding mobile phone application that we have developed with the name "*Color Witch*".

A nice feature of this application is that it also leeds users to the playgrounds of their childhood.

1.1 Game Description

The rules of the original game are simple. The game can be played by a minimum of two children, but it is certainly more fun when many players are participating. One of the player is the *witch* and her/his aim is to catch the other players. A session of the game starts with the witch deciding a color; from that moment on all the other players run away trying to find an object or a texture of the chosen color. If a player can touch the chosen color before the witch catches her/him, then that player is safe. On the contrary, if the witch catches a player before she/he can find the chosen color, the player becomes the new witch. If all players touches the chosen color before the witch can catch any of them, then the witch has to select another color and the game goes on.

Color Witch is clearly an outdoor game so that kids can run around looking for the color that can make them safe. It is a popular outdoor game and part of the fun is to choose colors which are not widespread in the environment where the kids are playing; for instance, green would not be a smart choice for a witch when playing in a park.

We have chosen to create a version of this game which can be played on mobile phones: a player is required to take a picture of an object/texture of the required color and the phone, once recognized the picture as valid will declare the player safe. Simple rules and modern technologies may be the right combination to success, especially when the technology required has a very large diffusion, as in our case. From a research point of view, Color Witch represents an interesting case study as its deployment includes some technological challenges such as, for instance, the interpretation of images taken with the phone camera. Moreover, this application shows how certain games that would not be playable or fun on a home PC/consoles applications can instead be even better if exploiting the right devices, i.e., mobile phones, thus allowing a free play experience [9]. Therefore, mobile games do not necessarily be just as a limited version of highly performing PC games; rather, they can add new segments to the gaming market.

Indeed, our digital game can be seen as specifically tailored for mobile devices, exploiting their peculiar features. Furthermore, it can enhance the fun of the original outdoor game by adding some new features as: (i) considering object identification instead of simple colors, and (ii) remote gaming with participants connected through the Internet. To demonstrate the feasibility of the project and of its benefits, we have created a proof-of-concept version of the game where two players can compete against each other through bluetooth connectivity. We discuss here issues related to the game design, communication protocols, and color recognition algorithm.

The rest of this paper is organized as follows. In Section II we analyze basic requirements for our application. Then, in Section III, we delve into design and implementation issues related to the project, whereas in Section IV we discuss several interesting directions for expanding the current work. Finally, Section V concludes this paper.

2. APPLICATION REQUIREMENTS

Before discussing the project design, we need to describe our digital game and to point out some basic requirements for the project; here, we present a list and a description of the main ones.

2.1 Game Mechanics

In our digital version of the Color Witch game, a player starts the game session and her/his mobile phone runs the witch program that has to choose a color. Other players can join the game with their own phones and receive on their display the information about which color has to be captured to be safe from the witch. The operation of capturing a color is performed through taking a picture of the object/texture of that color; in essence, a player has to take a picture which has a predominance of pixels of the chosen color (within a certain threshold). The game can be played in various ways; for instance, employing a timer; however, for this specific case and without loss of generality we have decided a simple rule: the player which first takes a picture containing the requested color, wins. Of course, this implies that the information about the picture just taken, or the picture itself, should be immediately transmitted to the witch in order to elect the winner; we have actually done this through bluetooth connectivity.

2.2 Game Modes

First, we have to decide which game modes are available.

1. **Single player:** a user may want to improve her/his ability on shooting the required colors (each phone camera can have different resolutions, different color sensitivity, etc.), in order to be the quicker and have more chances to win;

2. **Multi-player:** this is the general game mode involving more than one player, it can be split into the following two sub-modes:

   (a) Server: the *witch mode*, the player can start the game, and decides the color;

   (b) Client: all the other non-witch players.

The game modes are mapped onto a default client-server architecture: there is only one server (the witch); all the other players connect to the server to transmit their game events (e.g., information about the predominant color in a picture that the player has just taken).

2.3 Number of Players

In a real scenario Color Witch must have at least three players: the witch and two other players competing for survival. The greater the number of players, the funnier the game will be due to the increasing competition.

Being mostly a proof-of-concept, in this first implementation the application supports only the single player mode and two non-witch players in multi-player mode. Yet, this is currently sufficient for successful testing, and the subscription functionality can be easily extended to support a (greater) variable number of players.

2.4 Cameras Resolution Compatibility

Different phones are equipped with different cameras with various resolutions and color sensibilities. A player could take advantage of the situation and win more games by exploiting the fact that its camera allows an easier match of the predominant color in a picture just taken and the color chosen by the witch. Even the opposite can happen, where a player has a hard time to convince the game logic that the picture she/he has just taken actually includes

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8719
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8719

in a predominant way the color that was chosen by the witch. In both cases, we have to take care of unfairness by carefully flattening the camera outputs: pictures taken from every player will be reduced to a minimum common format for resolution and number of colors before processing; this way, the game logic will give even possibilities to all players for a successful shot.

3. DESIGN AND IMPLEMENTATION

3.1 General Architecture

The proposed application has been implemented using Java Micro Edition. In particular, the project makes use of a number of concurrent threads, each one dedicated to a specific task. To hold the code dedicated to each task, we have also defined some libraries. These libraries provide middleware interface APIs for the core to access basic functionalities.

- *Graphical user interface*. It manages user interactions and screen.
- *Color recognition*. This function is mainly in charge to identify the dominant color in a picture.
- *Media capturing*. It provides an interface to the camera hardware, this way the main program is not required to interface any external software provided by the phone vendor. The game execution flow lies in the thread implemented by this library: it keeps taking pictures until one of the player wins or the network connection is lost. This library coordinates directly with the one in charge for network management so as to send pictures and receive notification messages. There are three variants of media capturing depending on the game mode: single player, multi-player client, and multi-player server. In single player mode there is no network involvement, whereas in the other two modes the network is used to send information about a taken picture or to communicate the winner of the match.
- *Network management*. In this package connections with other phones are managed. The basic idea is to provide APIs to the application core to be independent by the network technology. Currently, the only supported technology is bluetooth but we are investigating the use of UMTS/GPRS and WiFi. This library takes care of connection set-up, tear-down, and packet exchange. When a player starts a new game session, a new instance of a server thread is created. This thread creates a Bluetooth connection and announce itself while waiting for players to join the game. Players looking for a game session create an instance of a client thread, which waits for bluetooth announces to join the game.

Other operations are managed by the main thread, including data synchronization and storage configuration.

The goal of the above structure is that of keeping all hardware- and interface-dependent functionalities separated by the application logic; this way porting to new phones and hardware platforms will be much easier.

3.2 Bluetooth Management

In this section we are going to focus on the bluetooth connection management; first, we describe the protocol and then we provide some implementation details.

Communication between phones is organized using message bundles composed by a command and an optional parameter. Serialized bundles are sent over the bluetooth connection. With this kind of approach we can easily transfer messages with a semi-structured semantic.

Even if it could be considered as quite simple, we deem that it could be useful to illustrate it. In essence, the communication protocol can be divided into two main phases:

1. a handshake phase;
2. a game phase.

In the handshake phase (see Fig. 1) two applications try to play together. In the picture, typewriter text is the message content and colon (`:`) separates the command from its optional parameter. As soon as the incoming player opens the connection a `WELCOME` message is sent by the server with the color to seek out as a parameter. The incoming player may reply with a `WANNAPLAY` message to register and start the game.

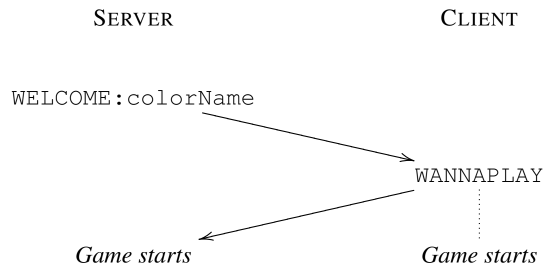

Figure 1: Handshake messages.

During the game phase, each application show to the player the output from the camera. Players keep taking pictures until winning the game or receiving a "game over" message. At a certain point, each player either loses or gets a correct color picture.

When the application is in game phase in a multi-player game mode, the network management thread is waiting for a bundle with the `YOULOSE` command. This command tells the receiver that the other player has already shot the required color.

As in many other phone-based games, the evaluation of the condition to win (the decision on whether the image contains the requested color or not) is performed by the client. Of course this architecture is open to cheating, but sending all pictures over the network and performing evaluation by the witch would use a lot of bandwidth and consume way to many energy. Moreover, calculation on the server side will limit the scalability of the game when engaging a greater number of players. Identification of the optimal trade-off between cheating possibility and energy consumption is still under investigation.

When the winning condition is met, (i.e., a picture with the chosen color has been taken), a message `IWON` is sent to the server. Currently, with the mobile phones we used in experiments and due to bluetooth's limited bandwidth, it is not possible to send the winning picture along with the `IWON` message quickly enough. The witch would have to wait for a time which – although quite limited – results unacceptable by users. For this reason, currently, only a notification bundle is sent back to the server.

Finally, in Fig. 2 we can see an UML state diagram with all the "high-level" application states. In this figure, the possible states for the application are described. Basically, there is an infinite loop in which the player chooses the game mode and has a match. At the end of the match, the player can decide whether to terminate the application or to play again immediately.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8719
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8719

3.3 Media and Colors Management

To decide the image's dominant color, we apply the following formula to the raster image:

$$\arg\max_{c \in C} \sum_{p \in P, p=c} p$$

where C is the set of colors and P is the image pixels set. In other words, the image dominant color is the one that covers the largest surface on the picture.

In the current implementation, the available colors are: black, white, red, green, blue, and yellow.

As already mentioned, there are considerable portability problems caused by different phone cameras.

3.4 Dominant Color Identification

An important function of the application is the identification of the image's dominant color.

Acquired images are represented in raster format: a matrix containing color information for each pixel. This information is initially stored in ARGB (Alpha, Red, Green, Blue) format and then we convert it into RGB (Red, Green, Blue) as the transparency feature provided by the Alpha channel is not useful for our purpose. Indeed, the ARGB format is a compulsory choice since the Java ME APIs manage images only in this format; yet, in order to compare the pixels with the chosen color, we have preferred for simplicity to ignore the alpha channel and to use the RGB image format. To further simplify and speed up color recognition we have adopted the HSV (Hue, Saturation, and Value of lightness) representation for the RGB color space. With HSV it is possible, for every color, to pinpoint bounds for its identification.

A possible representation of HSV is reported in Fig. 3. In the figure all possible hues are shown in the right column, whereas lightless and saturation scales are reported in the left square.

After associating a color to every pixel of a taken picture, it is possible to find the dominant color of the image by identifying which color is present in the picture with the greatest number of pixels, and then return it.

Creation of bounds for each color is an important point to choose the precision we want in color recognition. It is easy to see that, by giving small bounds, we increase the precision, but the game will be more difficult.

A big problem we have faced during development is related to light reflection on surfaces. Indeed, reflection hue is recognized as the white color thus cheating our software: an object/texture having a clearly dominant color, but a great light reflection, is often recognized as white. This effect depends on the position of the light, but also on poorly manufactured camera lenses; this way, different phones may have different color evaluations due to poor image quality. This problem can been partially solved taking off white from the list of selectable colors and increasing slightly the value of V before processing. Nevertheless, this approach would reduce choices for the witch, reducing the fun, and does not work in the same way on all mobile phones due to the ability of some camera to perform color correction.

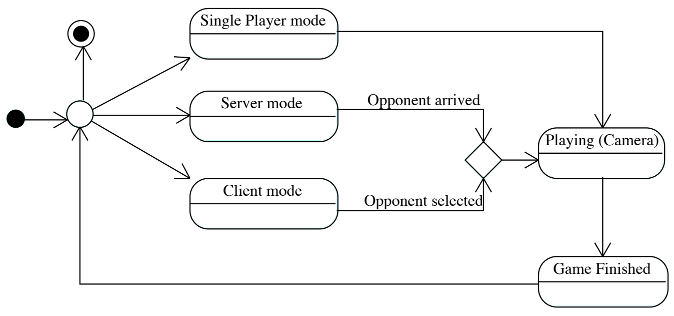

Figure 2: Application UML state diagram.

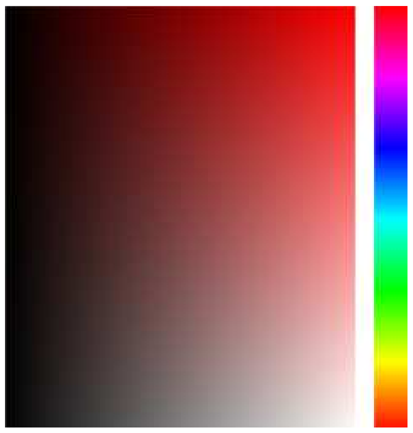

Figure 3: HSV representation for a reddish color.

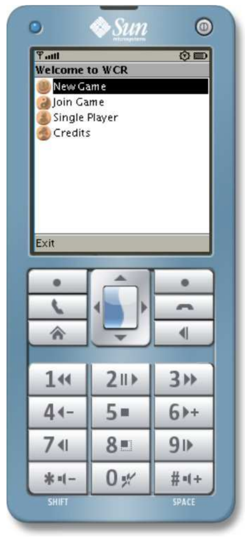

Figure 4: The application main menu, running on the SUN simulator.

3.5 Application Deployment

In Fig. 4 it is possible to see the application main menu running in the SUN simulator, which is built inside a NetBeans IDE project. Our application has been tested also on a couple of real phones: NOKIA N70 and 6085. In Fig. 5 a NOKIA 6085 running our application is shown. To stimulate contributions we decided to released the source code to the public domain using Source Forge [14].

4. ONGOING WORK AND EXTENSIONS

In this section we describe ongoing work about extending our game. Note that these are only a subset of all possible extensions: we deem the following points have a higher priority than others.

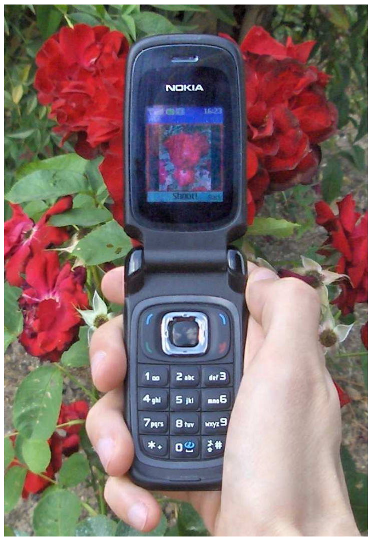

Figure 5: The application running on a NOKIA 6085.

4.1 Improved Colors Identification

We think the most important extension is the improvement of the dominant color identification. Currently, this task presents various issues; in particular, the identification of the most appropriate threshold for color identification. As explained in Section 3.4, the application identifies a color using some bounds check. A possible extension could involve the automatic identification of these values after a training session. During the training session the application learns the bound values which maximize the similarity between the real-object dominant color and the calculated one. This could also be a better way to avoid the unfair advantage/disadvantage for players with a better/worse camera, as mentioned in Section 2.4.

4.2 More-Than-Two-Players Mode

Another extension which can increase the appeal of our application is the implementation of a multi-player game mode where more than two players are allowed to participate. Indeed, it is well known that humans are social beings and this characteristic has been well captured by online game researchers that have devoted significant effort in developing scalable solutions [15]. Clearly, it would be very interesting to extend this application in order to allow more clients connected to the server. Even in this new scenario, we think to the server as the game witch, and to all the clients as peer players.

This new feature will involve added complexity in the network management, like client/server disconnections and messages forwarding. Indeed, to generalize this option, the game should provide the possibility for players to play even when remotely located with respect to each other. As an example, using 4G connectivity (the best available among Wi-Fi, bluetooth, UMTS, GPRS, etc.) players could participate virtually from everywhere improving the scalability and the fun of the game; or even building a community.

4.3 Game Scores and Web Leaderboard

An important part of the gaming experience is represented by the scoring function of the game. This is generally associated to the rewarding possibility for players to perceive, improve, and record their skills, and to compare their highest scores against each other. To facilitate the last point, a nice game service may be an on-line score sharing functionality. In any case, to determine a score, the first thing to do is to determine a score function. Possible options for this function could be:

- the time elapsed before shooting the correct color (the quickest the better);
- the number of game sessions won;
- a grade to the best picture given by the witch;

Whit the application able to collect the game score, it will also be possible to send it to an Internet website containing global and regional ranking. Score submission can be easily performed using the HTTP protocol.

4.4 Simon Says... a Shape!

The ultimate goal of this project is actually to extend the current game, moving out of the color realm. One possibility would be to ask players to take a picture where a pre-determined shape is present. This will change the game into an more interesting activity: it would still be used as a game, but also as a learning platform for kids or a rehabilitation cure for some mental disorders.

We envision two major difficulties along this path. The first problem is that, despite the fact shapes recognition can already be performed on a PC, we do not know whether the existing solutions would be energy efficient and fast enough for an average-level mobile phone. Probably, new solutions, specifically designed for a limited device have to be developed. The second problem is how to define an interface to let the witch *draw* a shape on the considered phones and be able to transfer that, possibly irregular, shape to the players.

5. CONCLUSIONS

Games represent a killer application for mobile phones. In this paper we have discussed the importance of providing games which are not just a tiny version of a PC game. Indeed, nowadays, a new generation of mobile games is possible by fully exploiting the unique features of current phones (e.g., camera, gyroscope, connectivity, GPS). To this aim, we have created a proof-of-concept application that shows how to use the mobile phone's camera and ubiquitous connectivity among players to digitally replicate and even enhance a popular outdoor game for kids.

Finally, we have also proposed several extensions to our game so as to improve its scalability, social inclusion, and computational intelligence.

6. REFERENCES

[1] Symbian foundation `http://www.symbian.org/`

[2] Android consortium official website `http://www.android.com/`

[3] M. Furini, "Mobile Games: What to Expect in the Near Future", in *Proc. of GAMEON Conference on Simulation and AI in Computer Games*, Bologna, Italy, Nov 2007.

[4] S. Ferretti, S. Mirri, M. Roccetti, C. Sermenghi, V. Conforti, "Managing First Response Medical Aids With An Altruistic Web Application", in *Proc. of the 3rd ICST/ACM/IEEE Pervasive Health 2009*, London, UK, Apr 2009.

[5] Apple iPhone "App Store" `http://www.apple.com/iphone/appstore/`

[6] *Mobile gaming grows to EUR 6 billion in 2006*, 2003 `http://cellular.co.za/news_2003/101503-mobile_gaming_grows_to_eur_6_bil.htm`

[7] K. Jegers, M. Wiberg, "Pervasive Gaming in the Everyday World", *Pervasive Computing, IEEE*, vol. 5, no. 1, pp. 78-85, Jan-Mar 2006.

[8] A. D. Cheok, K. H. Goh, W. Liu, F. Farbiz, S. W. Fong, S. L. Teo, Y. Li, X. Yang, "Human Pacman: A Mobile, Wide-area Entertainment System Based on Physical, Social, and Ubiquitous Computing", *Personal and Ubiquitous Computing, Springer London*, vol. 8, no. 2, pp. 71-81, May 2004.

[9] Regan L. Mandryk, Kori M. Inkpen, "Supporting Free Play in Ubiquitous Computer Games", in *Proc. of the Workshop on Designing Ubiquitous Computing Games (UbiComp 2001)*, Atlanta, GA, USA, Dec 2001.

[10] Martin J. Wells, *J2ME Game Programming*, 1st ed. Course Technology PTR, 2004.

[11] C. Bala Kumar, Paul Kline and Timothy J. Thompson, *Bluetooth Application Programming with the Java APIs*, 1st ed. Morgan Kaufmann, 2003.

[12] Rafael C. Gonzales and Richard E. Woods, *Digital Image Processing*, 3rd ed. Pearson Prentice Hall, 2008.

[13] Sun Microsystem, *Documentation for the Java Platform, Micro Edition (Java ME) and Java Card technologies*, `http://java.sun.com/javame/reference/apis.jsp`

[14] *Color Witch on Source Forge*, 2009 `http://sourceforge.net/projects/wcr/`

[15] A. Ploss, S. Wichmann, F. Glinka, S. Gorlatch, "From a Single- to Multi-Server Online Game: A Quake 3 Case Study Using RTF," in *Proc. of ACM Advances in Computer Entertainment (ACE 2008)*, Yokohama, Japan, Dec 2008.