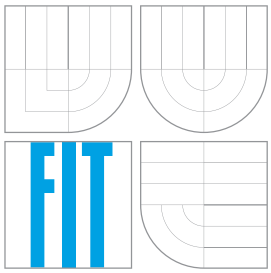
BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

BOARD GAME USER INTERFACE WITH A CAMERA

UŽIVATELSKÉ ROZHRANÍ S KAMEROU PRO DESKOVOU HRU

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR
AUTOR PRÁCE

DITA CIHLÁŘOVÁ

SUPERVISOR
VEDOUCÍ PRÁCE

prof. Dr. Ing. PAVEL ZEMČÍK

BRNO 2016

Brno University of Technology - Faculty of Information Technology

Department of Computer Graphics and Multimedia — Academic year 2015/2016

Bachelor's Thesis Specification

For: **Cihlářová Dita**
Branch of study: Information Technology
Title: **Board Game User Interface with a Camera**
Category: User Interfaces

Instructions for project work:

1. Study the literature concerning user interfaces utilizing cameras. Focus on the recognition of current status of a board game.

2. Study the ways to transmit images captured from a smartphone camera to a PC and the image processing techniques needed to identify the board game and the components on it. This includes the software tools required, such as OpenCV.

3. Choose a suitable board game and design a user interface with a smartphone camera.

4. Implement the user interface and demonstrate its functionality on the chosen board game. The application should observe the current status of the game and determine its next action when playing against the real player.

5. Evaluate the results obtained and assess the possibilities of future work.

Basic references:

- dle pokynů vedoucího

Requirements for the first semester:

- Body 1 až 3 zadání

Detailed formal specifications can be found at http://www.fit.vutbr.cz/info/szz/

The Bachelor's Thesis must define its purpose, describe a current state of the art, introduce the theoretical and technical background relevant to the problems solved, and specify what parts have been used from earlier projects or have been taken over from other sources.

Each student will hand-in printed as well as electronic versions of the technical report, an electronic version of the complete program documentation, program source files, and a functional hardware prototype sample if desired. The information in electronic form will be stored on a standard non-rewritable medium (CD-R, DVD-R, etc.) in formats common at the FIT. In order to allow regular handling, the medium will be securely attached to the printed report.

Supervisor: **Zemčík Pavel, prof. Dr. Ing.**, DCGM FIT BUT
Consultant: Debono Carl J., prof. Ing., Ph.D., UMALTA
Beginning of work: November 1, 2015
Date of delivery: May 18, 2016

L.S.

Jan Černocký
Associate Professor and Head of Department

Abstract

The aim of this work is to create a system that would be able to substitute a real opponent in a board game *Horse Races and Bets*. The game is observed by a camera that transmits images to a computer. The image is then processed and game objects are identified. After that, the application can analyse a current status of the game and decide about the next move of the artificial player. The decision-making algorithm is able to react to every game situation and thus play the game to the end. The application can be used by players that like to play *Horse Races and Bets*, but currently do not have an opponent, and for demonstrating the possibilities of computer vision.

Abstrakt

Cílem této práce je vytvořit systém, který bude schopen nahradit reálného protihráče ve hře Dostihy a sázky. Hra je snímána kamerou, ze které se obraz přenáší do počítače, kde probíhá zpracování obrazu a identifikace herních objektů. Dále aplikace analyzuje stav hry a rozhodne o svém následujícím tahu. Rozhodovací algoritmus je schopen reagovat na každou herní situaci a tudíž dohrát hru do konce. Aplikaci mohou využít hráči, jež rádi hrají Dostihy a sázky, ale chybí jim protihráč. Aplikace také může najít uplatnění jako demostrace možností počítačového vidění.

Keywords

user interface, image processing, object detection, camera, smartphone, Horse Races and Bets, Monopoly, board game, AI, FTP, OpenCV, ZBar, augmented reality

Klíčová slova

uživatelské rozhraní, zpracování obrazu, detekce objektu, kamera, smartphone, Dostihy a sázky, Monopoly, desková hra, UI, FTP, OpenCV, ZBar, rozšířená realita

Reference

CIHLÁŘOVÁ, Dita. *Board Game User Interface with a Camera*. Brno, 2016. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Zemčík Pavel, Debono Carl J..

Rozšířený abstrakt

Deskové hry se v dnešní době stávají znovu populárními. Jsou oblíbeným prostředkem k prolomení ledů ve skupině neznámých lidí i k upevnění vztahů v rodině nebo partě přátel. Rozvíjejí logické a strategické myšlení, jemnou motoriku a další schopnosti uplatnitelné v praktickém životě. Metodami jako kooperace, soupeření nebo obchodování podporují sociální interakci mezi hráči. Deskové hry jsou dnes na vysoké úrovni, a to jak nápady a hratelností, tak grafickým zpracováním. Přesto je možné je dále vylepšovat – například pomocí počítačové technologie.

Technologie může být v prostředí deskových her uplatněna mnoha způsoby, například k omezení rutinních činností, jednoduššímu seznámení s pravidly, zajímavějšímu hernímu prostředí, atd. V této práci je technologie využita k vytvoření umělého hráče, který dokáže hrát deskovou hru Dostihy a sázky.

Aby to bylo možné, musí mít umělý hráč k dispozici informace o stavu hry – jaké má kdo karty, na kterém políčku je hráčova figurka, kolik mu zbývá peněz, atd. Tyto informace lze získat s využitím kamery či fotoaparátu, který snímá obraz hry. Obraz se poté v počítači v reálném čase zpracuje a informace zde obsažené se analyzují. Tímto způsobem je umělý hráč schopen rozpoznat herní situaci.

Kromě komunikace skrze hru samotnou také probíhá komunikace pomocí grafického uživatelského rozhraní, které je navrženo jako chat. Aplikace se tak snaží co nejvíce napodobit hraní původní hry se skutečnou osobou.

Práce obsahuje studii a analýzu uživatelských rozhraní s užitím kamery či fotoaparátu a také přehled základních technik zpracování a přenosu obrazu, které jsou využitelné v této aplikaci. Kromě toho uvádí i nástroje použitelné pro uvedení těchto technik do praxe.

Board Game User Interface with a Camera

Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of prof. Dr. Ing. Pavel Zemčík from the Brno University of Technology, Faculty of Information Technology and prof. Ing. Carl James Debono Ph.D. from the University of Malta, Faculty of Information and Communication Technology. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Dita Cihlářová
July 27, 2016

Acknowledgements

I would like to thank prof. Dr. Ing. Pavel Zemčík and prof. Ing. Carl James Debono Ph.D. for their professional advices, helpfulness and great patience. I also want to express my gratitude to prof. Ing. Carl James Debono Ph.D. for helping to make my studies in Malta a valuable experience.

Contents

Chapter 1

Introduction

Despite a massive and successful videogame industry, board games are getting more popular again. They promote logical and strategic thinking, fine motor skills, decision making and many more skills applicable in everyday life. They also encourage rich social interaction by means such as cooperation, competition or trading. However, there are still many opportunities to improve these games. In this work, technological means are employed to create a system that is able to substitute an opponent in a board game.

The idea came from the author, who is fond of playing board games, but sometimes can not find anybody willing to play the game. For such people, an artificial opponent could be a convenient solution.

A real player interacts with the artificial opponent mainly by playing the board game in a traditional way. Therefore, the board game itself can be considered a significant part of user interface – a communication channel from user to computer. This approach is possible with a camera that continuously provides images of the game. These images are transferred to a computer, where the application processes the image and retrieves information about current game status. This way, an artificial player is able to „see“ what a real player would see. The information that players usually tell each other verbally can be passed to the application via a Graphical User Interface (GUI).

Concerning the opposite side of a user interface, computer-to-user communication, the information for the user is presented in adequate and attractive way via the GUI. This includes mainly information about the artificial player's current turn and also instructions for the user, for example where to move artificial player's game pieces.

One of the subjects of this thesis is to choose a suitable game for demonstration of the implemented user interface with a camera. A game called *Horse Races and Bets* was chosen for this purpose. It has a challenging design with many game objects (such as a gameboard, cards, game pieces and tokens). On the other hand, rules are not very complex and the game depends more on luck than strategy. This fact makes it possible to focus more on the user interface and image processing and less on the game logic.

The rest of this thesis is structured as follows: Chapter 2 introduces existing solutions concerning board game user interface with a camera and board game augmentation. This chapter also outlines the principles of *Horse Races and Bets* and shows several examples of this game's digitization. In Chapter 3, image processing principles that are relevant to this work are described, together with possible ways of image transmission. Chapter 4 analyses and evaluates state of the art described in previous chapters. It also provides a plan of work that is fundamental for future implementation, which is described in Chapter 5. Finally, Chapter 6 summarizes the results and assesses the possibilities of future work.

Chapter 2

Approaches to game augmentation and digitization

In this chapter, state of the art concerning game digitization is described. Due to the limited range of this thesis, only the most relevant information for the implemented solution is mentioned. The chapter starts in Section 2.1 with the pros and cons of board games and computer games. Then it continues in Section 2.2 by augmenting a board game with digital elements. Many techniques can be used for such augmentation, but this thesis focuses on usage of a camera, therefore Section 2.3 shows three different examples of camera-observed games.

Horse Races and Bets is the game that was chosen for demonstration of the implemented application. Its principles are outlined in Section 2.4. Existing digitized versions of *Horse Races and Bets* are summarized in Section 2.5.

2.1 Board games versus computer games

This section focuses on board games, computer games and the differences between these two. It provides a brief summary of pros and cons that could be useful for design of the final application. This summary was partially inspired by [9].

Board games, a specific type of tabletop games, are games played by moving pieces on a special gameboard. They have several important advantages. The most obvious is social interaction; people play board games because they find it a good way to spend time together. a board game can be played by people who know each other for a long time, like family and friends, or it can be used as an icebreaker among a group of complete strangers. This is possible thanks to game rules, which define the way of playing, but at the same time encourage the players to compete or cooperate, according to the nature of the game. This usually leads to rich social interaction and a great experience for the players. Another advantage is usually an appealing design. Players can appreciate a visual appearance of game accessories. The last advantage discussed are tangible game pieces. Players can feel more in control of a game when they can touch and move the game pieces by their own hands.

Computer games, on the other hand, tend to be less social. There is a number of multiplayer games available, many of which can be played by people in one room (so called LAN party). However, social interaction is still reduced due to the fact that every player has to look at his own display and keep his hands on a keyboard or mouse. This prevents

eye contact and hand gestures, which are very important parts of interpersonal communication. Another cons can be inflexible rules or a user interface, that can be harder to use e.g. for certain age groups. The biggest advantage of computer games is graphics. Modern computers have almost endless possibilities when it comes to creating various environments, worlds and characters. Graphics and cinematics can also be utilized for advanced storytelling, immersing a player into a game. Another positive aspect of computer games is that they are interactive and can support a player when needed, for example explaining rules at right time or providing a feedback for player's actions.

2.2 Augmented game environment

Augmented games stand in between of classic board games and computer games. Augmenting a game environment, as stated by [5], means that some supportive digital element is added to an already existing board game. The purpose of this element can be for example: more transparent gameplay (e.g. highlighting the parts important in a specific moment), keeping track of events and scores, report rules violations. More examples of how can an augmented game environment support players of a board game can be found in [5, p. 100]. This paper also offers guidelines for designing and implementing augmented game environments by means of pervasive computing technologies. The authors suggest a two-step process: *game flow virtualization* (modelling a game, its objects and rules) and *physical augmentation* (providing game objects with pervasive computing technologies elements). The guidelines for physical augmentation can be seen in appendix A. Finally, the guidelines are put to use in an example game, W41K, which is an augmented version of a complex board game Warhammer 40,000.

2.3 Board games observed by camera

The term Tangible Augmented Reality (TAR) refers to a board game mixed with auxiliary technology, typically a camera and a projector [7]. The pros and cons of three versions of the same game, Settlers of Catan, are examined in [7]: the original board game version, a TAR version, and a completely digitized version on an Apple iPad. The TAR version of the game consists of tangible game pieces, cards and dice, but a gameboard is projected on a table. The game is recorded by a camera, the image is processed and numbers on the dice are recognized. This can help players by highlighting the gameboard's areas connected with the numbers rolled. This solution employs OpenCV for image processing.

Another project deals with observing a game in a casino [4]. Several cameras observe a game table. The image is processed and cards are detected and identified with OpenCV methods. First, blobs of proper size are found. Then their edges and corners are identified. Finally, when a blob has all four corners detected, it is considered a card and a computer tries to identify it. This system is designed for common usage in casinos and serves for helping staff to check that players follow the rules.

A next possible reason for observing a game with a camera is to make a record of the game. Later, the record can be processed and used for an analysis of the game by players or their teachers. In [15], the author describes such a project. The author uses a camera on a tripod for taking photos of the game of chess or Go. After that, image processing is used for getting information about a current state of the game in the picture. The author describes his algorithm that can detect a gameboard, black and white stones and moves of

these stones. With this information, it is possible to make a simulation of the game.

2.4 Horse Races and Bets

Horse Races and Bets (originally *Dostihy a sázky*) is a Czech game based on a world-wide known game called Monopoly. The objective is to become the wealthiest player through buying, renting and selling horses, attending horse races with them and betting on horse races. a player who wants to win needs to analyse the situation, weigh the facts, invest his funds cleverly and also be lucky. In next two sections, the most important game principles are outlined. An emphasis is placed on aspects that can be useful for game status recognition.

Game description

A main component of Horse Races and Bets is a gameboard. It is a square shaped board with a diameter about forty centimetres. The gameboard is composed of 40 game fields with various purposes:

- 28 fields of properties,
- 6 event fields,
- 6 other specific fields.

On the property fields, including horses, trainers and services, a player can buy the property and later make profit from it. When a player comes to event field, either Chance or Finance, he has to draw a corresponding event card. There are also 6 other fields with their own specific rules: Start, Suspension, Carpark, Doping suspicion and Clinic (2x).

A next feature of the game are property cards. Every property card is associated with one game field. When a player steps on given field and the card is still free, he can buy it. When the card already has another owner, the player has to pay a fee to the owner. The property cards include:

- 22 horses,
- 4 trainers,
- 2 service cards (Stalls and Transport).

The second type of cards included in the game is an event card. Event cards are further divided to Finance cards and Chance cards, each associated with different game fields. However, their purpose is the same. When a player steps on a Finance field or a Chance field, he draws an appropriate card. The card contains instructions that the player has to follow, such as „pay 1000“ or „go three fields forward“ etc.

Game pieces, representing players on the gameboard, are another essential game element. Every player has one game piece distinguished from the others by its colour. a player moves his game piece forward between the game fields, according to a number that was rolled on a die.

Another necessary part of the game is money. The game comes with a set of notes of different values. Each of the seven types of notes can be distinguished from the others by

Figure 2.1: *Horse Races and Bets*, the game chosen for demonstration of the implemented solution.

its colour, pattern and number. In the beginning of a new game, every player gets 30,000 of game money. a player who loses all his money has to leave the game.

The last important game objects in this list are Races. Races are small circular tokens that can be placed on horse game fields to signalize that the horse participates in a race. There are two types of tokens: smaller yellow ones (Races) and bigger red ones (Major races).

Gameplay

Players take turns in a clockwise order; highest throw of a die starts a game. Each turn, a player rolls the die. If there is a number six on the die, the player rolls the die again and the results add up. The player then moves his game piece forward, given by the total number thrown. After moving, the player immediately performs an action connected with a field he came to.

During the game, players try to buy horses and other useful cards and make profit from them. The cards can be purchased from a bank for the price stated on the card or from other players for any price. The biggest profit comes from races and bets on races, therefore every player tries to buy races as soon as possible. In order to buy a race, a player needs to have a complete stable. a stable consists of two or three horses with the same background colour of their cards. Only in case that a player owns the whole stable, horses from that stable can participate in a race. The race is represented by a yellow token. Its cost is individual for every horse, according to its card. One horse can attend at most 4 races. When 4 races are purchased, a horse can participate in a major race, which is represented by a larger red token.

When a player comes to another player's race, he has to pay the owner a price stated on the card of the racing horse. However, the player can also make profit from races of other players by betting; before the player rolls a die, he can put some amount of money on a race in front of his game piece. If he then actually comes to that field, an owner of the race pays him ten times the bet amount. Otherwise, the owner takes the whole bet. a player is allowed to bet only if some of the player's horses participates in 3 or more races.

A player that can not pay his debts (has no money or property left) has to leave the game. The game continues until there is just one last player, who becomes a winner.[1]

2.5 Digitized Horse Races and Bets

Horse Races and Bets have been transformed into a computer game several times. The main goal was usually to preserve all principles and rules of the original game while providing the possibility to play with other players via the Internet. All the analyzed solutions are very similar to each other in functionality and also in graphical design. They all display a gameboard, additional game information and several buttons as control elements.

A product of a bachelor's thesis [10] is a fully functional digitized game that keeps all the original rules including trading between players and furthermore offers playing over a network. Another bachelor's thesis [1] is very similar, providing the same functionality with a slightly different user interface.

The last bachelor's thesis analysed [8] extends the game with an artificial intelligence, providing a possibility of „player vs. computer" mode. Finally, this feature is also included in a software product Dostihy 3000 deluxe from Nextcorp team.[2]

All the solutions introduced are to be further analyzed and discussed in Chapter 4.

2.6 Game augmentation and digitization summary

In this chapter, the concepts of board game, computer game and augmented game environment were described. The next part introduced three examples of a tabletop game observed by a camera. In last two sections, the principles of the chosen board game, Horse Races and Bets, were outlined along with several examples of the game's digitization.

[1] Complete rules in the Czech language can be found at: http://www.mojedino.cz//out/pictures/wysiwigpro/63116%208%

[2] More information at: http://dostihy.nextcorp.cz/

Chapter 3

Transmitting and analysing an image

In this chapter, chosen approaches concerning image processing are described. Due to the limited range of this thesis, only the most relevant information for the implemented solution is mentioned.

The implemented application needs to get information about current game status. A source of this information is an image taken by a smartphone camera. The image needs to be transmitted from the smartphone to a computer. Section 3.6 introduces some of the possibilities to do this.

It was also needed to study possible approaches to finding and identifying objects that could appear in the game. The term *image processing* is used for this task. Therefore, basic information about image is outlined in Section 3.1, along with two of basic, widely used image operations: noise removal and thresholding.

Game objects are usually designed to be easily recognizable by a human player. They are identifiable by their shape, size or colour. These aspects can also be helpful for automated computer-based recognition. Finding an object according to its shape is possible for example by applying edge detection, as described in Section 3.2 and analysing the result. More complex objects can be found by template matching, which is introduced in Section 3.3. Game objects that have specific colour can be identified using colour-based segmentation outlined in Section 3.4.

Some game objects are more difficult for recognition - e.g. cards, that are identifiable only by the small text printed on them. Such objects can be enhanced with some kind of marker that makes recognition easier. Section 3.5 introduces QR code, which can be conveniently used as such marker.

The theory introduced in following sections can be put to practice by a set of tools described in Section 3.7. Finally, this whole chapter is summarized in Section 3.8 to review the most important information.

3.1 Image representation and basic operations

Image is mathematically defined as a continuous function of two variables, as can be seen in Equation (3.1).

$$z = f(x, y) \tag{3.1}$$

Because image has limited size, it is possible to describe its domain of definition as a Cartesian product of two intervals [17].

$$f' : \langle i_{min}, i_{max} \rangle \times \langle j_{min}, j_{max} \rangle \rightarrow H \tag{3.2}$$

For Equation (3.2) applies a following constraint: $(i_{min}, i_{max}, j_{min}, j_{max}) \in \mathbb{Z}$. For a convenient representation in the computer, H is perceived mostly as a set of three values (for RGB model) or one value (light intensity in grayscale model), as described by Equation (3.3) and Equation (3.4).

$$H \in \langle 0, 255 \rangle \times \langle 0, 255 \rangle \times \langle 0, 255 \rangle \tag{3.3}$$

$$H \in \langle 0, 255 \rangle \tag{3.4}$$

Noise removal

Smoothing, also referred to as *blurring*, is a technique used in image processing for reducing noise in image. In signal theory, noise is described as a new information that was added during recording or transmitting a signal [17]. There are several methods that try to reduce the noise and thus restore the original image. However, many of these methods perceive noise just as a high-frequency information - similar for example to valid edges. Therefore, they have to find a way to analyse pixel's surroundings and decide whether the pixel belongs to original image or not [17]. The methods used for smoothing include for example Gaussian blur, Median blur or Bilateral filter [2].

Thresholding

Thresholding is a method commonly used for further reduction from grayscale images to monochromatic images. This method is based on comparing light intensity of every pixel with a given value, referred to as a threshold. The threshold can be either set manually or computed from processed image's known parameters, such as median of the pixel intensities [14]. Compared pixels with intensity lower than the threshold become black and pixels with higher intensity become white.

3.2 Edge detection

Edge detection is one of the most important ways to extract information from image. It is widely used to find objects of specific shape. An edge can be described as a sharp change in light intensity of adjacent pixels. However, the same thing can be said about noise. Therefore the common goal of all edge detection algorithms is to find as much real edges as possible and, at the same time, to leave out as much noise as possible.

Canny edge detector is a good example of an edge detection algorithm. It was introduced by John F. Canny in 1986 [3]. Since that time, many improvements have been created, but the basic principles stay the same, described as follows:

1. Apply a Gaussian filter to reduce noise.
2. Compute the gradient intensity representation of the image.

3. Apply non-maximum suppression that rejects pixels that are not part of the local maxima. The resulting image contains thin lines – candidate edges.

4. Track edges using hysteresis thresholding.

The last point is what makes Canny edge detector different from its predecessors [2]. Hysteresis thresholding uses two thresholds instead of one. If a pixel's gradient value is above the upper threshold, it is accepted as an edge and coloured white. If it is below the lower threshold, it is rejected and coloured black. If the gradient value falls between the two thresholds, it is accepted only when it is connected to another pixel whose gradient is above the upper threshold [2, 6].

3.3 Template matching

Edge detection can be very helpful when detecting simple geometric shapes, but it is not so convenient when it comes to more complex objects. Fortunately, there are other options. When there is a picture of desired object available, it is possible to use a technique called *template matching* to detect this object in a different image. Nowadays, there are many alternatives to this method or more modern techniques with similar purpose. However, template matching is an interesting method that is sufficient for this kind of project.

The sought-after picture is called template. One of the ways to find a template in an image is to „slide“ the template over the image and at each location, compute some measure which says how much the template matches the current subpicture. This measure is then compared to a threshold and it is decided whether the template exists in the picture [16].

3.4 Colour-based segmentation

Different colours used for creating an image are composed of several basic colours from the colour spectrum [17]. For example, a colour that can be seen on a screen is a result of combining three components: red, green and blue. They can be represented by a triple, whose components have values in range $\langle 0, 1\rangle$. However, this value is often presented as an integer in range $\langle 0, 255\rangle$, which is more convenient for mathematical operations and also for storing, because it can be stored in one byte.

The above described model is called *RGB colour space*. There are many other colour spaces, each suitable for some kind of usage. *HSV colour space* also uses three components: *hue*, *saturation* and *value* [17]. While saturation and value describe finer shades of a colour, hue is basically the component that distinguishes basic colours from each other, like green from red (see Table 3.1). This can be well utilized in simple colour-based segmentation [11].

Colour-based segmentation is one of many methods that can be used when searching for object that is distinguishable from the background by its colour. It works similarly to thresholding. A user has to provide an original image and two thresholds, for example two HSV values. Every pixel, whose value falls between those two thresholds, is considered a match and its value is set to 1. All other pixels are set to 0. This way, a monochromatic image is created. White regions represent regions of chosen colour in the original image and black regions represent everything else [11].

Colour	From	To
Blue	[110,100,100]	[130,255,255]
Green	[50,100,100]	[70,255,255]
Yellow	[20,100,100]	[30,255,255]
Red	[175,100,100] [0,100,100]	[179,255,255] [7,255,255]

Table 3.1: Examples of HSV values of basic colours, as represented in OpenCV. These values can be used as thresholds for colour-based image segmentation. Values of Hue component representing red colour can be found at both ends of the Hue range.

3.5 QR code

Quick Response (QR) code, defined by the ISO/IEC18004 standard, was developed as a modern type of barcode. While original barcode can hold just very limited amount of information, QR codes can contain up to 23 648 bits [13]. Another important advantage is error correction. QR code can have one of the four levels of error correction, while the highest level (marked by letter H) can restore up to 30 % of encoded data [13].

QR code consists from modules (dark squares) organized on light background in a square matrix. These modules represent data, version and format information and specific patterns for easy recognition of position, size and inclination of the code. A structure of a QR code can be seen in Figure 3.1.

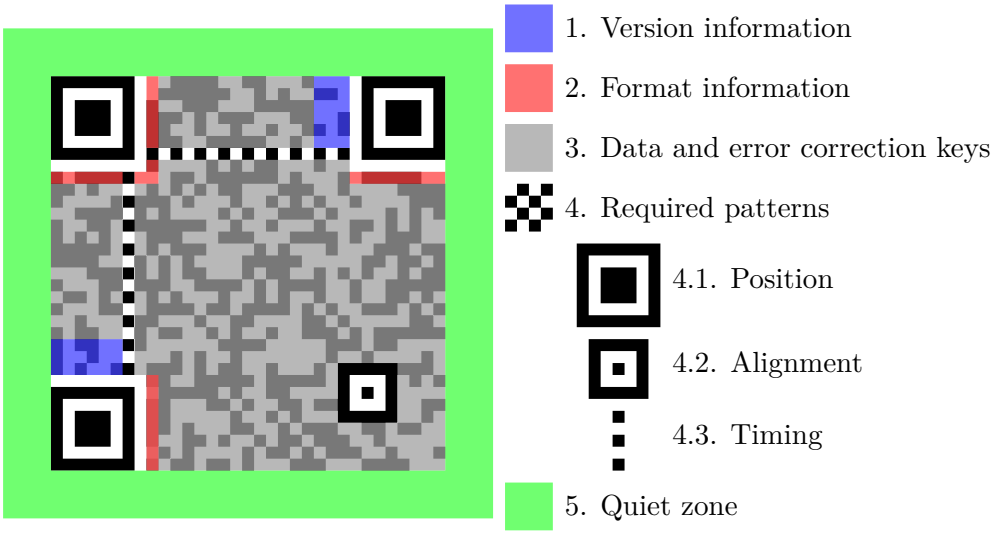


Figure 3.1: QR code structure. Adopted from: [18]

QR code can have various size depending on how much data it stores and what is the level of error correction. The size of a QR code can be determined from the version [13].

3.6 Transmission from smartphone to computer

There are many ways of wireless data transmission from smartphone to computer: Bluetooth, NFC, FTP, LAN, Internet, etc. However, the implemented application needs a way

that is stable, easy to set up for a user and can be conveniently automated. Because of limited space of this thesis, only the way analysed as the most suitable is described: FTP.

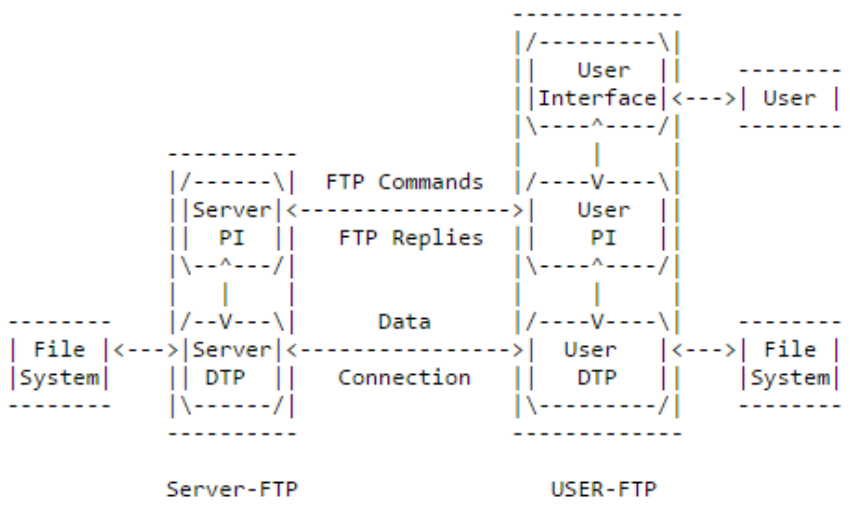


Figure 3.2: The diagram of FTP usage, adopted from the original RFC (found at [12]). The Data connection does not have to exists all the time.

File Transfer Protocol (FTP) is a protocol that aims to transfer data reliably and efficiently. It is meant to be used mainly by programs [12]. As seen in Figure 3.2, a user initiates a connection via user's protocol interpreter. A server replies via its own protocol interpreter. This is called *control connection* and its aim is to establish and maintain *data connection*, which is used for transferring required files [12].

Smartphones can establish a FTP server too. There are many application that provide a user-friendly solution for this issue, some of them even free.

3.7 Tools

This section introduces the tools that were analysed and utilized in the implemented application. One tool is needed for several image processing operations. Second for QR code detection and recognition. And another for FTP connection.

OpenCV

OpenCV (Open Source Computer Vision Library)[1] is an open source computer vision and machine learning software library. It features more than 2500 classic and modern algorithms. It provides many possibilities of usage, from recognizing objects and faces, classifying human actions in videos and extracting 3D models of objects, to recognizing scenery and establishing markers to overlay it with augmented reality, and many more.

It provides C++, C, Python, Java and MATLAB interfaces and supports operating systems Windows, Linux, Android and Mac OS.

[1] More information at: http://opencv.org/

ZBar

ZBar is an open source software designed for finding and decoding several types of bar-codes, including QR codes. It can be used on its own via included GUI and command line programs, or utilized in scripts or programs via some of its interfaces (Python, Perl, C++).

It works on many platforms including Linux, Windows and iPhone. Due to its high performance, ZBar is suitable also for real-time scanning from video streams.[2]

FTPServer and ftplib

The tools chosen for FTP connection between a smartphone and a computer are FTPServer and ftplib. FTPServer[3] is a free application for the Android operating system. It is able to set up an FTP server, using port, login and password of user's choice. It has a user-friendly design and can be easily controlled, but it also provides detailed logs concerning the course of connection and files transmission, which is very useful during debugging.

The tool used on client's side is called *ftplib*. It is a python module that provides many operations useful for client's side of FTP connection. It is embodied in The Python Standard Library.[4]

3.8 Summary of image transmission and analysis

In this chapter, the principles of chosen image processing techniques were explained. These techniques include smoothing, thresholding, edge detection, colour-based segmentation and template matching. They were selected because of their high applicability in game objects recognition.

This chapter also came with an applicable way of image transmission from smartphone to computer (FTP) and with solution for enhancing game objects with auxiliary markers (QR codes). Finally, the chosen tools for executing above described solutions were introduced.

[2] More information at: http://zbar.sourceforge.net/index.html
[3] More information at: https://sites.google.com/site/andreasliebigapps/ftpserver
[4] More information at: https://docs.python.org/2.7/library/ftplib.html

Chapter 4

Analysis and work plan

The solutions described in [chapter 2] are further analysed in this chapter. Their possible influence on the implemented application is discussed. According to this analysis, further work plan is outlined.

4.1 Evaluation of the current solutions

This section broadens Chapter 2 with an evaluation of the described solutions. A stress is put on the aspects that are important for designing this product. A design process is divided into three parts: obtaining the image, artificial player's algorithm and graphical user interface.

Obtaining the image

Concerning the games mentioned in Section 2.3, there are different approaches for obtaining an image of a current game status. In [4], the authors use several cameras for viewing the game from different angles. This can prove useful in commercial environment when there is no space for mistakes and thus the system has to be very solid. However, most of the games that are meant mainly for entertainment does not need so complex and expensive solution. In this case, one camera seems to be enough.

Different studies differ in the way of placing and holding the camera. In [7], it seems from photos of the experiment that a camera is placed next to the overhead mounted projector. Both are presumably fixed to the ceiling. This solution can provide very good images, but it takes a lot of time and effort to set up an environment for a game and after that it is not mobile.

On the contrary, in [15], a camera is held by a tripod which is placed on adjacent table. This solution seems to be highly versatile and comfortable for users, but the images are taken from a very sharp angle, so it could prove harder to process such image. It seems that this approach was possible in the experiment because a chess gameboard is simple and can be easily reconstructed even from a bad quality image.

Artificial player‘s algorithm

In [8], the author implemented an Artificial Intelligence (AI) that should be able to play against a real player. He claims that he used Minimax algorithm with nondeterministic

modification (so called Expectiminimax). However, he later describes it as nearly unusable, because it is very slow.

It also seems that this algorithm is not chosen well, because Expectiminimax deals with zero-sum games, which is for example Backgammon, but not Horse Races and Bets. The author does not explain what his final solution was, but Minimax algorithm does not seem suitable for this type of game.

The *Dostihy 3000 deluxe* project[1] offers a game with an artificial player, too. However, this software is not open-source, so it could not be found out how the algorithm had been implemented.

Graphical user interface

All digitized games described in Section 2.5 serve well their purpose. They try to preserve the *look and feel* of the original game. Their GUIs imitate the game so they seem familiar to users who have played the board game. The principles of the game also remain the same, so the players are able to play instantly, without learning new rules. However, the implemented application is not a digitized game, therefore a different approach is required.

The principle of the implemented application is very close to the idea of Augmented game environment, described in Section 2.2. Some of the guidelines addressing this issue (see Appendix A) can be applied to a GUI of the implemented solution: especially points 3, 4, 8 and 12. According to these advices, the focus should remain on the game and the interaction itself, not on the technology; therefore, the technology used should be unobtrusive and secondary user interfaces should be minimized. Player's access to information should be simple and efficient. A player should receive the right information at the right time.

4.2 Utilization of the analysis

The information gathered from analysis of current solutions is further used for designing the application. The three parts of the design process (obtaining the image, artificial player's algorithm and graphical user interface) are further discussed, this time with a more specific outcome.

Obtaining the image

A lot of people do not have a digital camera at home, because modern smartphones offer a good quality of images and video. These devices can also provide several possibilities of transferring an image to a computer. Therefore, a smartphone was chosen as a device that will be obtaining the images of the game.

A smartphone should be held above the gameboard, so the image will not be distorted. Such position can be achieved by a tripod, etc. The device must be placed on some elevated position in order to have the whole gameboard and cards in one image.

Artificial player‘s algorithm

The Minimax algorithm from [8] was not considered a good solution. Furthermore, because the game is quite specific, an own algorithm is required. The algorithm should be fully functional in all game situations, allowing the game to be played to the end. However, an

[1] More information at: http://dostihy.nextcorp.cz/

advanced AI is not a main purpose of this work. Therefore, the algorithm does not need to be very sophisticated.

Graphical user interface

According to the guidelines mentioned in Section 4.1, the goal is to keep the GUI lightweight, simple and unobtrusive. A real player should primarily look at the gameboard. He should look at GUI just at times when he would normally look at the other player – for example in situation when the other player is performing his move. Therefore, it is possible to say that the GUI should not display the actual game, but rather imitate another real player, providing only information that the other real player would provide by his words and actions.

This is one of the steps required to preserve the feeling of the original game. Such effect is considered desirable, because it could make the application more attractive for people who have played and liked the original game. This kind of users could probably constitute the main target group.

4.3 Custom implementation requirements

This section summarizes the outcome of the previous section. A custom specification of the implemented application is outlined. This specification suggest ten requirements in order to provide a better idea of what and how should be implemented.

1. The application should observe the current status of the board game Horse Races and Bets and determine its next action when playing against the real player.

2. The application should use one smartphone camera positioned above the gameboard to obtain the image of the game.

3. The application should process the image, using OpenCV, and detect objects that are necessary for obtaining basic information about a current game status.

4. The application will be implemented in the Python language, which, among other features, provides well-documented wrappers for OpenCV functions.

5. The design of the board game can be slightly modified to facilitate image processing. However, the game has to remain playable even when the implemented application is not used. Furthermore, all game objects have to remain recognizable by users that have played the original game before.

6. The application should have a fully functional gameplay algorithm that is able to react to every game situation, allowing the game to be played to the end.

7. The algorithm should follow the same rules as the original game. The principles of the original game should be preserved as much as possible.

8. A GUI of the application should be lightweight, simple and unobtrusive, while providing all the necessary information from artificial player and from the game environment (e.g. error messages).

9. The application should not end until it is ended by a user. In case that there is an error that makes it impossible to continue (e.g. wrong or missing camera input), the user should be asked to either fix it or close the application.

10. The application should contribute to the feeling of the original game as much as possible.

4.4 Summary of analysis and work plan

In this chapter, three different tasks that need to be solved were discussed: obtaining the image, artificial player's algorithm and graphical user interface. State of the art concerning these tasks was thoroughly analysed. Finally, further work plan was outlined.

Chapter 5

Implementation

This chapter explains the structure and implementation of the application. The implemented solution needs to complete five consequent basic steps:

1. Take a picture of a game,
2. transfer the image to a computer,
3. process the image and update data,
4. run a game algorithm,
5. interact with a player.

These steps are to be individually discussed in following part of the thesis. Section 5.1 covers the way of taking a photo of the game. In Section 5.2, transmission of the image from a smartphone to a computer is described. Section 5.3 outlines image processing techniques that were utilized for game objects recognition. It also describes modifications in look of the original game, that facilitate the image processing.

Section 5.4 is dedicated to description of algorithm for one turn of the game. A strategy of the Artificial Player (AP) is outlined. Section 5.5 explains the way that the computer can interact with the user and vice versa.

In Section 5.6, the modular structure of the application is described. Section 5.7 provides information about installation, usage and testing of the implemented application.

5.1 Obtaining an image

Image is to be obtained via a camera integrated in a smartphone. The image needs to be available primarily at the start of each AP's turn. This can be achieved by two ways: either take a photo on demand, or take photos periodically during the whole game.

The first option seems a little more reasonable, but in fact proved more difficult to achieve. The author wanted for this task to utilize an existing application, but have not found a suitable product that would be able to take commands from a computer and thus take a photo on demand. However, such application might be created as a further extension to the implemented solution.

The second option is to take pictures continuously. This solution's advantage is that it does not need any intervention from a computer. A new image is always available, the

computer just needs to take the latest one. So called *time-lapse* applications are ideal for this task.

Tina Time-lapse[1] is such application. It is available for free on Google Play shop. It can take unlimited amount of photos in adjustable quality. Due to the non-restricted amount of photos taken by the application, it is crucial to delete the old images regularly. In the implemented application, all images are removed at the end of each turn.

Importantly, a delay between individual shots is also adjustable. It was, after several experiments, set to two seconds. This is considered a sufficient reaction time for the implemented application to get a new image of the game. One second delay was also tested, but it occasionally resulted to freezing of Tina Time-lapse, supposedly because of insufficient performance of the used smartphone. Due to this experience, it is possible to expect that on weaker device, the two-second delay might also cause this kind of problem. However, this can be easily solved by increasing the delay to acceptable amount. This can result in prolongation of AP's turns, but the game is still well playable.

Figure 5.1: Game setup.

The smartphone running the Tina Time-lapse application is to be fixed in a tripod that is set up next to the gameboard, as seen in Figure 5.1. An image is therefore taken from above and contains the whole game.

[1]More information at: http://wesselrossing.nl/articles/tina_time-lapse

5.2 Image transmission

FTP has been chosen as a suitable option for transmission of images from a smartphone to a computer. It proved to be fast and reliable. The connection is realized by two tools: the FTPServer application on the side of smartphone and the ftplib python module on the client side. More information about these tools can be found in Section 3.7.

The FTPServer application can be easily configured by a user - he can set a port, login and password. The user then needs to provide these information, along with an appropriate IP address, to the implemented application. No further maintenance is needed.

For this solution to work, the smartphone and the computer need to be in the same network. This network can be a WLAN, but there is even more convenient solution available for many smartphones: they can make their own network by the Hotspot function. This proved to be very useful and reliable.

5.3 Image processing

After successful transmission of the image to the computer, the image needs to be processed and crucial game objects are identified. The methods described in Chapter 3 are used for these tasks. Further details about the implemented methods can be found in the source code on attached CD. The chosen tool for putting these techniques into practice is OpenCV (see Section 3.7).

The application needs to keep track of several pieces of game information in order for AP to be able to make decisions. This information include:

1. Location of the gameboard,
2. cards that the real player and the AP have in their inventories,
3. amount of tokens on individual horse fields,
4. current event card, when needed,
5. real player's and AP's money.

There are few other game objects that are important, but do not have to be identified. For example, an information concerning the location of AP's game piece is needed. However, AP is always able to calculate its position from previous information, therefore the real game piece does not have to be recognized in the image. A die is another example. The AP does not have to rely on a physical die because it uses its own virtual die.

Modification of the original game

In order to facilitate the image processing, some game components were slightly changed to meet the requirements of the employed computer vision algorithms. The adjusted game can be seen in Figure 5.1. The most visible change is absence of AP's money. The AP keeps track of its money digitally.

Another notable modification is that all game cards are enhanced with a QR code. These QR codes contain data needed for computer to identify the card. The last, subtle change are new race tokens. The original tokens were designed in several shades of yellow or red. The new tokens have the color united to one shade, which is more convenient for colour-based segmentation.

Gameboard recognition

A gameboard of *Horse Races and Bets* can be described as the largest square in the image. A way to find its coordinates is to detect the edges in the image (see Section 3.2) and then find such edges that constitute a square. Squares that are smaller than a threshold are rejected, which helps to get rid of smaller square-shaped objects like fields, etc.

The original image's resolution is reduced, so the detection does not take so much time. The result are coordinates of gameboard's corners in the smaller image.

Inventory recognition

The adjusted game is designed for two players: the real player and the AP. Each of them have their cards placed visibly by one side of the gameboard. The coordinates of the gameboard are already known for the reduced size of image. However, card recognition needs bigger resolution, so the coordinates found in small image are used for derivation of the coordinates corresponding with the original image. Thus, the original image can be easily divided into three parts, the vertical borders of the gameboard being the dividing lines. Then the outer parts of the image represent players' inventories.

These inventories can be consequently searched for game cards. Each card is marked with a unique QR code. The ZBar tool is able to find and decode all the codes existing in a given image. In each QR code, there is a string encoded that identifies the card. Strings that are not recognized as valid cards are rejected, so the recognition is resistant to random objects with codes placed in the scene.

Event card recognition

An event card contains special instructions, that have to be accomplished by a player that drew it. It comes into game when the player steps on *chance* or *finance* field. In this moment, a corresponding card have to be drawn. If the current player is the AP, then the real player is asked to draw a card and place it on the gameboard.

After that, the application tries to get the new image and identify the card in it. It works in the same way as inventory recognition, but the card is searched for in the middle part of the image - the gameboard. There should be only one card at time. Therefore, more or less than one QR code detected is considered an invalid state and the user is asked to fix the scene. The data in the QR code are further verified to find out whether the code really belongs to an event card.

Token recognition

Concerning the circular tokens called *races*, the application needs not only to detect them, but also to count their occurrence on different fields. An interesting and quick method for this task includes *colour-based segmentation* (see Section 3.4).

First, it is necessary to define the pixels that are occupied by individual fields. A pixel map was created for this purpose, so for each field, there is a list of pixels belonging to it. The second task is to decide, whether some pixels of a field are occupied by a race. The race have a specific colour that is different from background colour of all the fields. In case of small races, the colour is bright yellow, major races have bright red.

Therefore, the colour-based segmentation is used, first with these thresholds for yellow colour: [20,100,100] and [30,255,255]. The result is an image where all yellow pixels are

set to 1 and all other to 0. Because it is known that none of the fields had yellow colour before, all pixels in all fields should be set to 0 (black); therefore the pixels that are set to 1 (white) have to be a part of a yellow race. A ratio of black to white pixels in one field determines, how large area is occupied by yellow tokens. From this information, it can be counted how many races are there on the field.

The same process is conducted for red tokens, except for there are two pairs of thresholds applied. This is due to the fact the Hue value of red colour can be found at both ends of the Hue scale. At one end, it represents shades of red colour that are closer to orange, at the other end, the shade is closer to magenta colour. For exact values of the thresholds, see Section 3.4.

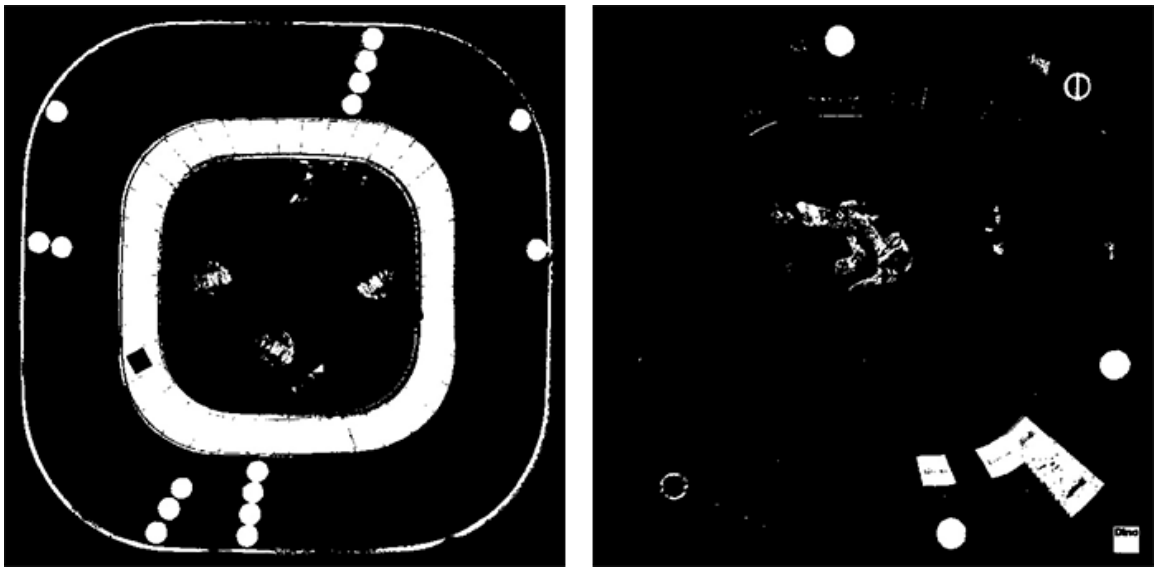


Figure 5.2: Gameboard segmentation based on yellow colour (left) and red colour (right). Races are clearly visible now.

Money recognition

Money in the original game consists of notes of different colours and values. That are features that could be used for recognition. However, the main problem is that a player can often have a big amount of money and has the notes arranged in a deck, so it is impossible to tell how many notes exactly are inside this deck.

This was the main reason why the author decided to omit money recognition from the image and instead keep track of AP's money digitally. The AP has no information about opponent's money, but as long as AP's strategy does not need to be highly sophisticated, this is not a problem.

However, during the game, a situation can occur in which the AP's money need to be adjusted during the real player's turn, so the AP would not notice it. Therefore, there is an option in the application for the real player to increase or decrease the AP's money. This way, the AP always has an information about its current state of money.

5.4 Game algorithm

After initial settings, the whole game consists of turns. Every time when it is the AP's turn the same algorithm is executed. The scheme of this algorithm can be seen in Appendix B. However, the most important parts are summarized as follows:

1. Update cards in player‘s inventory,
2. check for races ahead, possibly bet,
3. roll a die and move accordingly,
4. play according to the new current field,
5. check the money left.

First, a new image needs to be obtained and processed. From the acquired information, the game model of player's inventory is updated. After that, the possibility of placing bets is considered. The bets can be placed only under several conditions that have to be checked. If the AP meets the requirements, the application proceeds to token recognition. Only the horses reachable by a valid roll of the die are analysed due to greater amount of time needed for this operation. If any of these horses belongs to the opponent and possesses races, the AP decides how much money it bets.

Next stage is to roll a die and move to the corresponding field. The movement solution needs to take account of two extra situations:

- A player rolls number 6 two times. This means going immediately to Suspension.
- A player moves through the Start field. Therefore the player needs to receive 4000.

After the AP makes a move, it ends on some field. Every field has its own action or possibilities for a player. A player may be obliged to do something, pay or wait, or he has the opportunity to buy some property. After the decision is resolved, the AP's turn comes to last phase. The application checks, whether the AP still has enough money to continue playing. If yes, the AP's turn ends; if not, the game ends with the real player as a winner.

Artificial player's strategy

The AP's simple strategy is, apart from all the game's rules, based on few principles:

- If it is possible to buy a property or a race, buy it.
- If you have not enough money for a payment, sell property; begin from the least valuable to you.
- Betting: if you have enough money to place a bet with possible profit more than 1000, do place the bet.

This strategy is very simple and straightforward; for more enjoyable gameplay, it might be improved. One of the key points would be for the AP to focus on obtaining a whole stable; concerning the current state, the AP is prone to buying random horses until it runs out of money and then never having enough money to buy the missing horses to complete the full stable. This improvement is not a part of the assignment, but might be implemented as a further extension.

5.5 Interaction with real player

The interaction between the real player and the AP is realized via a Graphical User Interface (GUI). It is implemented in PyQt5[2]. The GUI needs to have several specific elements:

- A communication channel for displaying messages and instructions from the AP,
- the AP's money counter, with a possibility to edit the money,
- a way to notify the AP that it is its turn,
- a way to notify the AP that its instruction has been carried out.

A very important requirement is also that the GUI should be pleasant to look at and fun to work with. Taking these requirements in account, along with point 8 from Section 4.3 („lightweight, simple and unobtrusive"), a graphical user interface was designed (see Figure 5.3).

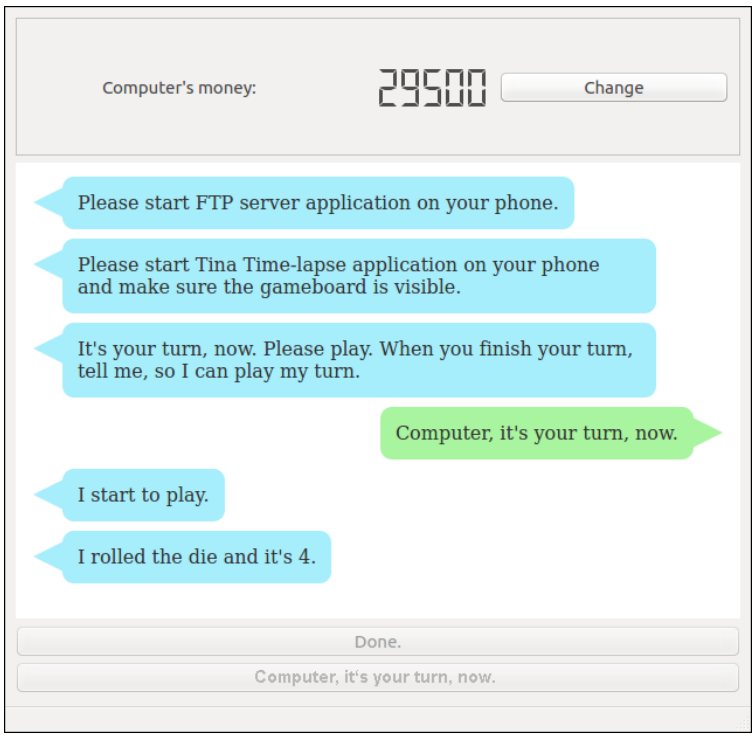


Figure 5.3: The graphical user interface designed for the implemented application.

This solution imitates a chat, increasing the impression that the user plays with another person. The user's messages are displayed on the right, while the AP's are on the left, just like it is common in SMS communication or on Facebook. Both players are also distinguished by a colour of message bubbles.

The AP sends two kinds of messages. First, the information about its turn: what number was on the die, what property was bought or sold, etc. Second, the instructions for the player, e.g. „Draw an event card for me.".

[2]More information at: https://www.riverbankcomputing.com/software/pyqt/intro

The user is able to send different two types of messages: *done* for confirmation of accomplished instruction and *your turn* to notify the AP that it can begin its turn. These messages are sent by simply clicking buttons, but are also displayed in the chat so it seems that the user has typed them on his own. This way, the communication looks smooth and natural.

Error messages

There are messages that need to be displayed to the user, but are not part of the chat. This includes announcing errors that need to be fixed by the user, e.g. lost FTP connection, missing image or unidentifiable scene. Such message is displayed in a modal window that urge the user to react and then confirm that the issue was solved. After the confirmation, the modal window disappears and the application continues in the chat.

5.6 Software structure

The backend of the application is structured into five modules. The *image_provider* module secures the FTP connection and all the operations needed to work with an FTP server. The *image_processing* module covers the implementation of methods utilizing OpenCV to work with an image. The *data_provider* module takes advantage of these methods to provide higher-level image processing operations which output data for model. The *model* module stores this data in structures resembling real game objects. It also provides methods connected with this data. Finally, the *hrabap* module (derived from a name of the application - *Horse Races and Bets Artificial Player*) serves to initialize and control the whole application.

A frontend is covered by the *UI* subpackage. There are modules defining classes that are responsible for *view* of all the application's windows. These classes are further employed in *hrabap*, where the behaviour of the windows is defined.

5.7 Installation, usage and testing

The implemented application can be installed according to the instructions that can be found in a file called README.md on the enclosed CD. For using it, the user needs: the board game *Horse Races and Bets* from the Dino Toys company, a smartphone and a tripod.

The smartphone must be fastened to the tripod approximately about half a meter above the game. The smartphone need to be running two applications: FTPServer and Tina Time-lapse. The implemented application gives user the instruction concerning running these mobile applications.

The application has been thoroughly tested with the following setup:

- A notebook running Ubuntu 16.04 LTS in Virtualbox,
- a smartphone running Android 4.1.2.

The tests were periodically carried out during the whole development. In the last stage, the whole solution was tested with the board game itself. Several bugs were discovered and fixed thanks to the testing.

Because this work deals with a user interface, it would be convenient to conduct a user testing. However, user testing is out of range of this work, so only the principles are to be outlined.

A user would be asked to setup and play the game. Then it would be observed whether he can do these tasks naturally, without confusion. The user would also provide a feedback about his impression from the application and the gameplay itself, and whether it resembles the original game. This could result in useful findings that could be utilized for further improvement of the application.

5.8 Implementation summary

In this chapter, the implementation of *Horse Races and Bets Artificial Player* was outlined. It was discussed how an image of the game is captured and transmitted, how the image processing and game algorithm work and how the application interacts with the user. Finally, the application's structure was outlined along with basic description of functionality of individual modules.

Chapter 6

Conclusion

The aim of this thesis was to design and implement an application whose user interface would utilize a board game and a camera. This goal was fulfilled. The implemented application can play the game, using images that are being continuously taken by a smartphone. A computer obtains a new image via an FTP connection at the start of each turn. The image is processed with operations that employ the OpenCV library and the ZBar tool. According to current game status analysed from the image, the Artificial Player (AP) decides about its next move and gives player instructions where to move its game piece, etc.

In order to design and implement such application, it was necessary to study the literature concerning user interfaces utilizing cameras. It was also needed to choose a board game and study current digital solutions dealing with this specific game. Another important prerequisite was the knowledge of techniques and tools usable for image transmission and recognition of game objects in the image. All these topics were studied by the author and described in this thesis. The state of the art was consequently analysed and evaluated.

The main goal of the user interface was to preserve the authentic experience from the original board game. The user should feel like playing the original *Horse Races and Bets*, as usual, with an opponent that is just not physically present. Therefore, the interaction with the user is ensured by a Graphical User Interface (GUI) that resembles a chat. The user communicates with the AP by using this chat and of course by playing the game - the AP „sees“ what the current status of the game is, just like a human opponent would see it.

Another important issue that needed to be solved was to find a compromise between two desirable states: preserving the look of the original game vs. facilitating image processing by adjusting the appearance of some game objects. This problem was successfully solved by three unobtrusive modifications that are very helpful for game objects recognition, but keep the original appearance of the game. All objects are still recognizable by the user and have not lost any informational value.

The application is to be used primarily by people that know the original game and like to play it, but at the moment do not have an opponent. Besides, the research, resources and findings can be useful to developers that want to employ computer vision for recognition of a board game or some different situation with known objects and rules.

Concerning future work, there are several possibilities of further development of this project. One of them might be to make the image processing more robust, so it would be able to cope with worse light conditions, etc. Furthermore, it could be interesting to improve the artificial player's strategy. That is also a prerequisite for another feature that could enhance the gameplay: trading between players.

Bibliography

[1] Eva Bergerová. "Desková hra Dostihy a sázky s podporou hraní po síti". Bachelor's thesis. The Czech Republic: Palacký University Olomouc, 2009. URL: http://library.upol.cz/arl-upol/cs/csg/?repo=upolrepo&key=47449024126.

[2] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. Sebastopol, California: O'Reilly Media, Inc., 2008.

[3] John Canny. "A computational approach to edge detection". In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), pp. 679–698.

[4] P. Gururajan and M. Gandhi. *Gaming object recognition*. US Patent App. 11/381,473. Apr. 2007. URL: https://www.google.com/patents/US20070077987.

[5] Steve Hinske and Marc Langheinrich. "W41K: Digitally Augmenting Traditional Game Environments". In: *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*. TEI '09. Cambridge, United Kingdom: ACM, 2009, pp. 99–106. ISBN: 978-1-60558-493-5. DOI: 10.1145/1517664.1517691. URL: http://doi.acm.org/10.1145/1517664.1517691.

[6] Jamie Hutton and Ben Dowling. *Computer Vision Demonstration Website*. Online. University of Southampton, 2005. URL: http://users.ecs.soton.ac.uk/msn/book/new_demo/ (visited on 07/21/2016).

[7] Jessica Wai Yan Ip. "An Augmented Reality Prototype for Investigating Tangible and Virtual Components in a Gaming Environment". PhD thesis. McGill University, 2011. URL: http://srl.mcgill.ca/publications/thesis/2011-MASTER-Ip.pdf.

[8] Adam Kulhánek. "Vývoj deskových her na platformě Java". Bachelor's thesis. Brno. The Czech Republic: Masaryk University, 2007. URL: http://is.muni.cz/th/98812/fi_b/Bakalarska_Prace (visited on 07/18/2016).

[9] Martin Nygaard et al. *Digital Matador*. 2009. URL: http://vbn.aau.dk/ws/files/18906395/Digital_Matador.pdf.

[10] Marek Očenáš. "GUI pro deskovou hru Dostihy a sázky". Bachelor's thesis. The Czech Republic: FIT VUT v Brně, 2011. URL: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=117709.

[11] *OpenCV: Changing Colorspaces*. Online. OpenCV community, 2016. URL: http://docs.opencv.org/master/df/d9d/tutorial_py_colorspaces.html#gsc.tab=0 (visited on 07/20/2016).

[12] J. Postel and J. Reynolds. *File Transfer Protocol*. Tech. rep. 959. Updated by RFCs 2228, 2640, 2773, 3659, 5797, 7151. Oct. 1985. URL: http://www.ietf.org/rfc/rfc959.txt.

[13] *QRcode.com*. Online. Denso Wave Inc., 2015. URL: http://www.qrcode.com/ (visited on 02/04/2016).

[14] Adrian Rosebrock. *Zero-parameter, automatic Canny edge detection with Python and OpenCV*. Online. PyImageSearch, Apr. 2015. URL: http://www.pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/ (visited on 07/18/2016).

[15] S. Scher, R. Crabb, and J. Davis. "Making real games virtual: Tracking board game pieces". In: *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. Dec. 2008, pp. 1–4. DOI: 10.1109/ICPR.2008.4761307.

[16] G. J. Vanderbrug and A. Rosenfeld. "Two-Stage Template Matching". In: *IEEE Transactions on Computers* C-26.4 (Apr. 1977), pp. 384–393. ISSN: 0018-9340. DOI: 10.1109/TC.1977.1674847.

[17] Jiří Žára et al. *Moderní počítačová grafika*. The Czech Republic: Computer Press, a.s. Brno, 2004. ISBN: 8025104540.

[18] Zephyris. *QR code structure*. Image under GNU Free Documentation License. Wikimedia Commons. 2011. URL: https://commons.wikimedia.org/wiki/File:QR_Code_Structure_Example_2.svg (visited on 07/22/2016).

Acronyms

AI Artificial Intelligence.

AP Artificial Player.

FTP File Transfer Protocol.

GUI Graphical User Interface.

LAN Local Area Network.

NFC Near Field Communication.

QR Quick Response.

TAR Tangible Augmented Reality.

Appendices

List of Appendices

Appendix A

Physical augmentation guidelines

This is the list of advices for physical augmentation as described in [5].

1. The technological enhancement should have an added value.
2. The supported actions and tasks need to be clearly specified.
3. The focus should remain on the game and the interaction itself, not on the technology.
4. Technology integration should be done in a way that is unobtrusive, if not completely invisible.
5. The game should still be playable (in the "traditional" way) even if technology is switched off or not working.
6. Design and implementation should be tightly coupled.
7. The technology should be reliable, durable, and safe.
8. Players should receive simple and efficient access to information. Feedback should be immediate and continuous.
9. The added technology should support the high dynamics of the game environments.
10. Development should follow an iterative process, including rapid prototyping and testing.
11. The operation of the integrated technology should be as maintenance-free as possible.
12. Secondary user interfaces should be minimized.

Concerning this thesis, these guidelines are further analysed in Section 4.1.

Appendix B

Algorithm of one turn