Puzzle Game

Matjaž Kosmač, Andrej Krota, Borut Batagelj
Faculty of Computer and Information Science, University of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia
E-mail: matt.kosmach@gmail.com , slofroggy@gmail.com , borut.batagelj@fri.uni-lj.si

Abstract - *The idea of the project was build at the University of Ljubljana, Faculty of Computer and Information Science, in scope of the course Communication Methods, in collaboration with Academy of Fine Arts and Design. It's main goal was to build an interactive system between a user and a canvas, using only a projector, infra-red (IR) transmitter and IR camera. In other words, user will be able to play some game with IR transmitter, which will be recognized by IR camera, with the main window of the application displayed on the canvas. The game implemented in the project is a puzzle and it's just for the purpose of showing, how the system can interact with it's surroundings. Projector projects the image on to the canvas and the player standing in front of the canvas is choosing the particles of puzzle he wants to exchange. Game has more different puzzle images to be solved, each composed of different number of particles. Number of puzzle particles grows, when the puzzle is solved. The most significant part of the project was to develop a method that recognizes the position of the IR transmitter in the image acquired from the IR camera. We will describe used methods and the difficulties we came across. We will also describe all elements needed, from the software used to hardware. The aplication can be transformed into more complex interactive system.*

Keywords – *Infra-red (IR) , Camera, Canvas, Interaction*

1. INTRODUCTION

Our main goal is to make interactive system, directed by IR transmitter, which allows us to interact with the canvas. The happening in front of the canvas (position of the IR transmitter), is recognized by the IR camera Logitech QuickCam. As the IR transmitter we are using remote control, which can be easily found anywhere and is accessible to anyone. To demonstrate how the interaction works, we built a puzzle game. Our colleagues from the Academy of Fine Arts and Design in Ljubljana suggested few games, such as puzzle, game Arkanoid and implementation of program Paint. Since some remote controls generate a short lasting IR light, our IR transmitter is not convenient for playing the game Arcanoid. The well known program Paint was already implemented by some of the other students. These are the reasons why we decided to build the puzzle game.

The main window of the application is composed of the puzzle, image we are composing, button for the next picture and the image from the camera just for help. The main window of the puzzle game is shown in Fig. 1.

The idea of capturing image from the camera and projecting the main window on the canvas is the following. On one side of the room there are projector and the camera. Projector projects image and camera captures image from the canvas, which is on the other side of the room. In front of the canvas (or behind canvas as shown in Fig. 2., if the canvas is permeable to the IR light) stands player and with remote control chooses the particle of puzzles, he wants to exchange. Two consecutively chosen particles will then be exchanged.

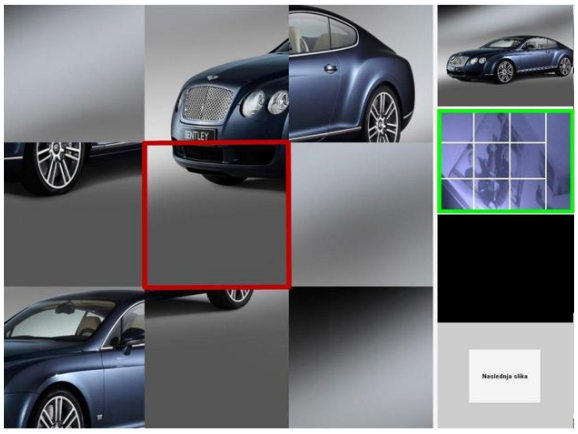

Fig. 1. Main window of the Puzzle game

Fig. 2. The setting of the camera, canvas and projector

2. USAGE

The puzzle we are trying to solve is displayed in the main window of the application. In the main window there is also an image we are trying to solve, image from the camera, and indicator, which is a sign, which allows us to select next particle that will be exchanged.

The player stands in front of the canvas where the application window is displayed. He moves the IR transmitter to the particle he wants to pick, stands still for a moment until the indicator appears (image from camera is bordered green), and then with pressing any button on the remote control produces the IR light signal. Selected particle is bordered red. The two following particles are exchanged. If we made a mistake by choosing the first particle to be exchanged, we select the area outside of the puzzle, and the system resets the selected particle. When we successfully solve the puzzle, the system gives us a new puzzle image to solve, with more paticles each time. At the beginning the puzzle is composed of 9 particles, the following puzzles are composed of 16 or 25 particles.

3. SOFTWARE AND HARDWARE

System requires IR camera, IR transmitter, projector and computer with the application Puzzle game. Computer also requires camera drivers and graphics libraries.

3.1. Hardware

Hardware can lead to extra costs in case of lack of some of the listed devices. System requires the following hardware:

3.1.1. Logitech QuickCam

A normal web camera must be modified, so that it can detect IR light [5]. System works better if we use IR camera and remote control, which is IR transmitter. Used camera has display resolution of 320×240 and detects waves with wavelength from 700 to 1000 nm. Size of main window must be a multiple of image size, aquired from the camera. Our main window is 4 times bigger than the camera's image. In this case, every pixel in camera's image represents 4 pixels in the main window.

3.1.2. IR transmitter (remote control)

We need IR light for selecting particle of puzzle or to make some other action. With other words, the place where IR light appears is the place of our cursor. It is similar how the computer mouse works. In analogy with the mouse, where the IR light appears is the *x* and *y* position of our cursor and the event is a click.

3.1.3. Projector

The projector projects main window onto the canvas. The user stands in front of the canvas and selects particles to be exchanged. Camera must capture only the region within the canvas, where the main window of the application is projected as shown in Fig. 2. Camera must be calibrated in order to detect only the happening in the main window of the application.

3.2. Software

Software cannot lead to extra costs, because we used open source libraries and drivers. Drivers can be usually found on the internet. Application is written in C++ and is platform independent.

3.2.1. Logitech QuickCam Drivers

It is necessary to install drivers for the used camera. They are usually enclosed to the camera, alternatively can be also downloaded from the internet [6].

3.2.1. IVT 1.3.2. library (Integrating Vision Toolkit)

The library enables the system to work with graphics. It is platform independent, written in programming language C++, therefore one of the quickest libraries in this field. Initial idea was to use SIFT method for detecting shapes, which demands quick execution of operations. Eventually it turned out that SIFT method is not the best way for detecting shapes in our project. On the contrary IVT library turned out as a great library and suitable for our task. The IVT library is also available on the internet [7].

3.3. Other equipment

3.3.1. Canvas

White canvas is the place where projector projects main window of the application. On the same side of the canvas are the camera and the projector (the projector above the camera), and on the other side is the user. If the canvas is not permeable to the IR light, the user must be on the same side as the camera and the projector, right in front of the canvas.

3.3.2. Extra light source

The camera is modified so that it detects IR light. If the room is not bright enough, there are not enough waves with wavelength from 700 to 1000 nm and the camera cannot detect waves with greater

or lower wavelengths. If there is not enough light in the room, the image from the camera is not clear enough, that's way it becomes dotted. To reduce this effect, we use an extra light source, pointed at the camera, which increases the amount of white light.

4. ALGORITHMS

The most important algorithm is shape recognition algorithm. We must somehow detect position of white light.

The happening in front of the canvas is sometimes unpredictable. For example if somebody opens the light or if somebody moves fast in front of the camera. In these cases the system could act unpredictably. We need a way to decrease the impact of such happening. This is the reason why we added the demand for decreasing the movement before the system acts on happening in front of the camera. If camera detects a lot of movement, the system simply ignores happening in front of the camera and does nothing. Otherwise it starts to look for the position of the IR transmitter in the image gained from the IR camera.
The importance of other algorithms is secondary.

4.1. Detecting the position of the IR light

First, we intended to use SIFT (Scale-Invariant Feature Transform) method, which can detect objects by finding edges in the picture. It turns out, that the method is inappropriate for two reasons. If the lighting of the room is insufficient, picture from camera is not clear (it means that the picture is dotted), and SIFT consequently detects too many edges. This problem can be reduced with appropriate lighting. Second problem derives from the shape of the remote control's light. Remote control's light has a shape of a circle. Since SIFT method detect edges, it is not the best method for detecting IR light.

Not such elegant but effective way of detecting IR light is comparing two successive frames (images) acquired from the camera. We must simply subtract two RGB values of two frames, every pixel in each frame. If there is a difference between RGB values of pixel, we turn pixel white, otherwise black. The new image will have it's pixels black where no motion was detected, and will have it's pixels white where the motion was detected, as can be seen in Fig. 3. This method also reduces the noise on a frame, acquired from the camera, and simplifies the recognition of IR light. By comparing two successive frames, we managed to decrease the unpredictable behaviour, but added new restriction to the system. Before selecting a particle that player wants to exchange, he must wait until the indicator (green light) appears, and that is when camera perceives no motion. Using histograms, we can easily find out, if there is no motion in front of the canvas. Our image has few white pixels. When the indicator appears, or with other words when there is no movement in front of the canvas, we expect that the player will press the button on the remote control, leading to the appearance of white pixels in our image. This is the only time our algorithm is trying to detect white circle object. The easiest way to recognize circle objects is by using the Pythagorean theorem. If d is the radius of circle and $P_c(x_c,y_c)$ is the center with it's coordinates, then the pixel $P(x,y)$ is inside the circle, if the following equation is true:

$$d^2 \leq (x_c-x)^2 + (y_c - y)^2 \quad (1)$$

If the pixel is white and all of it's neighbours within the distance of d are also white, then the pixel is the center of the white circle and also the center of the IR light. The value d, which is used in the system, is 25. This means that radius of the white circle must be at least 25. The IR transmitter in distance from 3 to 5 meters from the IR camera generates a circle-shaped light with radius grater than 25 pixels and the system can detect it. Center of the white circle is the center of the IR transmitter. The puzzle particle in the same position as the center of the circle is the selected particle.

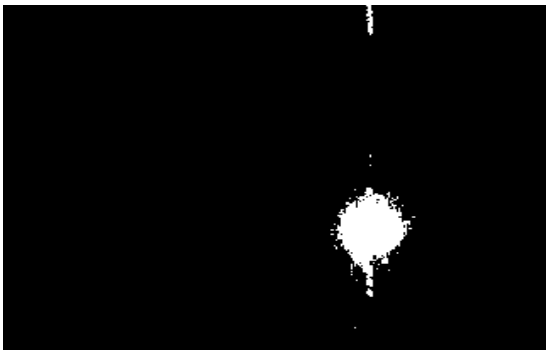

Fig. 3. Comparison of two successive frames assigns a black pixel if there is no motion, a white pixel where these is motion.

Fig. 4. Example of recognizing the shape and color of IR light produced by the remote control

4.2. Mixing the particles of the image

Mixing the particles of the puzzle is implemented using random number generator. With random generator we create random number *i* and *j*, $0 \leq i,j \leq n-1$, where *n* is the number of particles in our puzzle. The numbers *i* and *j* are the indexes of two particles, that will be exchanged. If we repeat this procedure many times, we will get particles randomly mixed. The data type for keeping the position of the particle in the puzzle is not necessary a table of images, but can be also a table of integers. If the image is divided into *n* particles, the indexes of each particle from left top to right bottom go from *0* to *n-1*. For example, if the second number in the table of integers is 4, it means that second particle contains the fourth particle of the original image. The random generator in programming language C++ generates a number from 0 to 32767. We need to mix particles as shown below.

```
for r = 1 to numberOfMixings:

    firstN = randomNumber(0, numberOfParticles)

    secondN = randomNumber(0, numberOfParticles)

    tempIndex= tableOfPictureInt[firstN]

    tableOfPictureInt[firstN] = tableOfPictureInt[secondN]

    tableOfPictureInt[secondN] = tempIndex
```

Fig. 4. Pseudoalgorithm for mixing particles

4.3. Other algorithms

Other algorithms are less significant. It's worth mentioning that number of particles is defined by the display resolution of the camera. Our camera has the resolution of 320×240. The puzzle takes the size of 240×240, the remaining (80×240) is used for user help. The size of the main windows must also be a multiple of the camera's image size. Our main window therefore has the resolution of 1280×960, 960×960 for the puzzle and 320×960 for help.

4. CONCLUSION

The main goal of the project was to build a system, which allows the interaction between the system and the canvas. The system can also be upgraded into more complex system. As a demonstration we build a game puzzle. Interaction promotes IR camera and remote control, if there is no demand for totally accurate system. System acts more reliably and correctly, if there is no demand for totally accurate border between the puzzle particles and demand for totally accurate position of the IR transmitter. In our case, borders of different particles of the puzzle are not defined completely accurate. This means that the system ignores the IR light if it appears near border between two particles. The IR light is relevant for our system only if it clearly appears in one of the regions, not between more regions. In this case, the system can work correctly and reliably.

ACKNOWLEDGEMENT

We would like to thank assistant professor Franc Solina for his help, advices and contribution to this work.

REFERENCES

[1] Azad, P., Dillmann, P. IVT - Integrating Vision Toolkit.
http://ivt.sourceforge.net
(last visited: December 28 2008)

[2] Wang, A.J.A. Camera-projector-based interactive game development.
http://portal.acm.org/citation.cfm?id=1233397
(last visited: January 15 2009)

[3] SourceForge.net: Sliding Puzzle 2x (27.1.2008). SourceForge.net.
http://sourceforge.net/project/screenshots.php?group_id=210020 (last visited: January 15 2009)

[4] SourceForge.net: LabVIEW – PuzzleGame (26.8.2006). SourceForge.net
http://sourceforge.net/projects/labview-puzzle
(last visited: January 15 2009)

[5] How to Make a Webcam Into an Infrared Camera: 6 steps (with video) – wikiHow
http://www.wikihow.com/Make-a-Webcam-Into-an-Infrared-Camera (last visited: May 17 2009)

[6] Logitech QuickCam Web driver – Logitech Camera Drivers
http://www.camera-drivers.com/drivers/275/275495.htm
(last visited: May 17 2009)

[7] IVT – Integrating Vision Toolkit
http://ivt.sourceforge.net (last visited: May 17 2009)