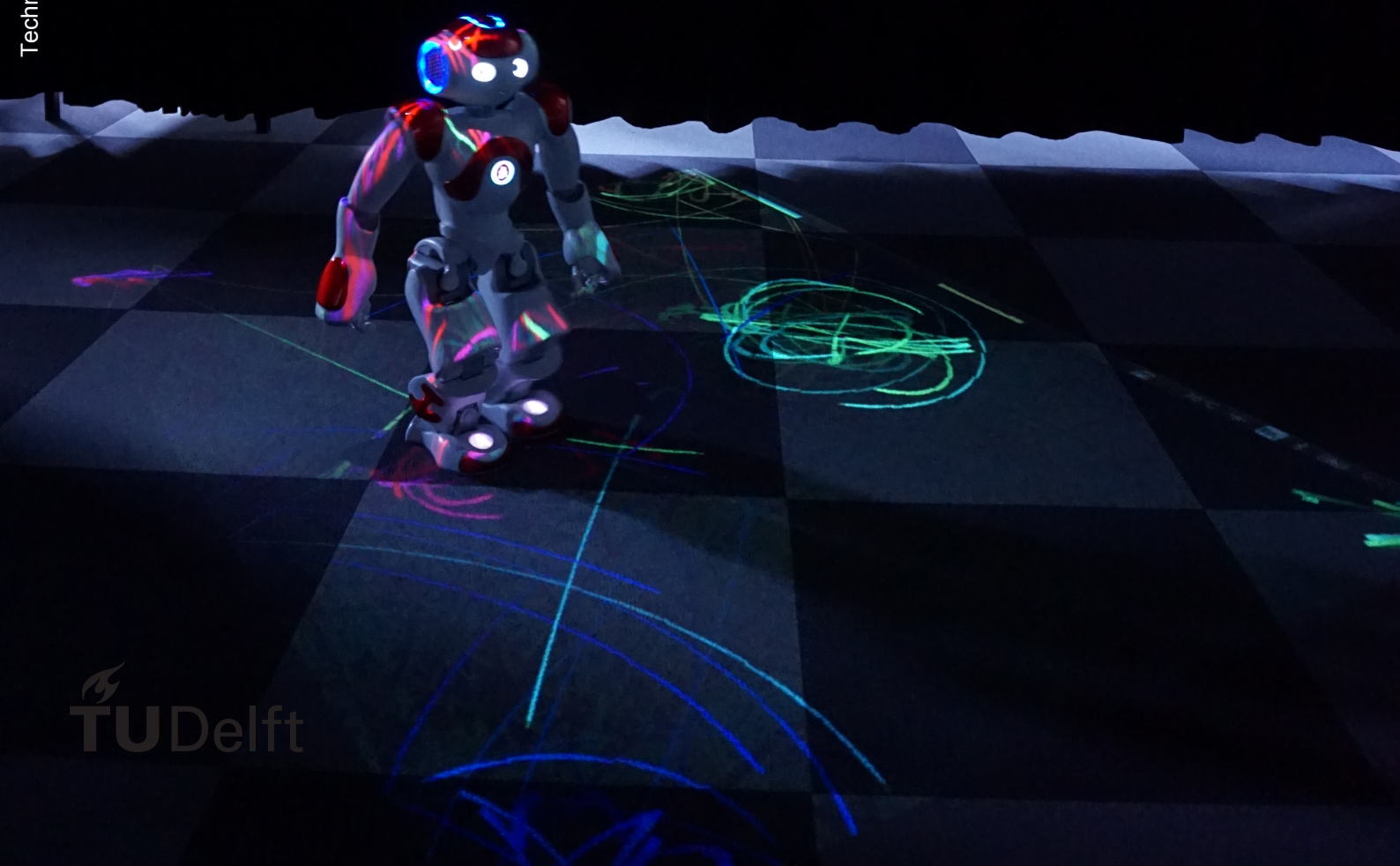
I.M.O.V.E.

Bachelor Thesis - Final Report

Marie Kegeleers
Gerbert Van Nieuwaal
Wouter Posdijk

Technische Universiteit Delft



TU Delft

z

I. M.O.V.E.

Bachelor Thesis - Final Report

by

Marie Kegeleers
Gerbert Van Nieuwaal
Wouter Posdijk

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University Of Technology
June 2016

Supervisors: Dr. ir. R. Bidarra, TU Delft
N. Salamon, TU Delft, Coach
BEP coordinator, TU Delft, Bachelor Project Coordinator

An electronic version of this report is available at http://repository.tudelft.nl/.

TU Delft

Preface

This report describes the process of the bachelor end project created by Marie Kegeleers, Gerbert Van Nieuwaal and Wouter Posdijk. It contains the problem description, problem analysis, research, development process, design choices, implementation details, testing process and concluding evaluation of the development of the project the past ten weeks.

This project was commissioned by the TU Delft Computer Graphics and Visualization Group. We were instructed to design and build an interactive installation for crowds in public spaces using people's movements and projections.

We would like to thank everyone who supported our project by providing us with all the tools and locations we needed: Ruud de Jong, TU Delft EEMCS technical support, Philippe van der Pal, TU Delft process coordinator CEH, and Paul Riem, TU Delft Aula technical support coordinator.

Finally, a special thanks goes out to our coach Nestor Salamon from the Computer Graphics and Visualization group for his guidance and help throughout the project and our client Rafael Bidarra from the Computer Graphics and Visualization group for his enthusiasm, efforts and constant feedback.

Marie Kegeleers
Gerbert Van Nieuwaal
Wouter Posdijk

Delft, June 2016

Summary

In our modern world of today, there are a lot of public locations with large open spaces where all people do is wait or pass through. Some examples are train stations, airports and waiting lines at theme parks. Even though people in these locations are surrounded by many other individuals, there is no social interaction and generally a gloomy atmosphere. I. M.O.V.E. is designed to change this. This interactive system based on motion tracking aims to entertain people using projections by motivating them to move around, explore and, most importantly, interact with each other.

To begin this project, after defining and analyzing the problem, research was done on the subject of interaction design to discover more about how people could be attracted and encouraged to participate, how to detect people, interactive systems that already existed, the necessary hardware and how to generate fun and attractive graphics. To ensure a fluent development of the project, the development process was established beforehand describing all tools used for collaboration, planning and general organization.

Based on the software design, I. M.O.V.E. was divided into three parts. The first part was calibration. Because the circumstances in which this system has to run are very diverse and unpredictable due to, for example, different projector and camera positions, the system has to be calibrated to be able to work properly on a specific location. The next part is people detection which extracts locations from people walking on the scene. The final part is the scene which generates art based on people's locations and movements. All the possible events in the scene and their details are described in the system design.

The system was well tested on different locations and machines. Each individual part was tested using specific methods and many integration tests were done to test the whole system and to optimize it for the final installation. An evaluation was done to describe the quality of the final product which includes a discussion and recommendations.

Contents

1

Introduction

People in public spaces tend to keep to themselves. They tend to stay within their comfort zone and do only that which they aim to do: take a break, wait for something or someone, or get to their destination. Often people are stressed out and moving too fast, and they need to settle down and be entertained for a moment. That is what we aim to do with I. M.O.V.E.

I. M.O.V.E. stands for Interactive Moment Of Visual Expression. With this project we aim to pull people out of their comfort zone and let them show their true, creative colors to the people around them. By letting people control projected art and allowing them to discover secrets through teamwork, we want to inspire them to be themselves, connect with other people, and turn a moment of rush or boredom into a moment of peace and entertainment.

After the description and analysis of our problem in chapter 2, the report will cover all the topics we researched during the first two weeks in chapter 3. After the research phase, the development process of the project started. All our methods regarding planning, development and organization will be discussed in chapter 4. When the proper research has been done and a general planning and development process have been established, the next step would be to design the system itself before starting the implementation. Chapter 5 describes this design and explains how the system behaves and what scenarios occur in different situations. This includes concrete choices about visuals and interaction events. Based on this design we can set up the software and start the implementation. I. M.O.V.E. is divided into three main parts which will all be discussed in depth in chapter 6 along with a few words about hardware and an introduction to the structure of the software explaining the software design. The parts described in the implementation chapter as well as the whole system need to be well tested. Because the system is separated into these independent parts, each part and the whole system require their own testing methods which will be described in chapter 7, which covers everything about the testing process. Also mentioned in this chapter are the different installations we deployed, the test results and solutions for encountered problems during testing. Finally, we will reflect on our methods, progress, experience and the final product in the evaluation chapter, chapter 8. We will evaluate our project through a discussion, reflection and conclusion and end with some recommendations.

2

Problem Description

2.1. Requirements

The goal of the project is to let passerby-crowd generate, interact with and control displayed projective art. Below are listed the requirements of the project.

- Location with passerby-crowd and an empty floor for the projection of at least 4m x 8m.
- Temporal setup of projective art on location.
- Crowd should be able tointeract with the projective art.
- Individuals should be able to interact with each other using the projective art.
- Changing locations of crowds/individuals must be tracked.
- Displayed projective art should be generated and controlled by individuals' movements.
- The art must not add more constraints than physical constraints. E.g. constrain or force movement to make the art work or require a smartphone app.

2.2. Problem Definition

The TU Delft Computer Graphics and Visualization Group came up with the idea to develop a system that could entertain passer-by crowd in an interactive way and translated this into a project. The goal of this project is to enable a large number of individuals to contribute to real-time procedural art using their movements. The analysis of the crowds' movements should involve image processing techniques applicable to a real-time video stream and the generated art is supposed to be shown using projection and should be generated and controlled by people's movements. The final result should be an interactive system that combines real-time analysis of video content from a crowded scene with an efficient transformation into an artistic projection stream to some output scene. Some challenges include developing real-time methods for generating art from data on people's actual, constantly changing location, using crowd analytics for artistic purposes and letting different people's interactions with the art intermix with each other in a creative way.

2.3. Problem Analysis

Because the project description is not very constrictive, it demands us to be creative and to decide ourselves what kind of system to design. This includes decisions about placement of the camera and the projector, what kind of art will be projected and how to make people interact with the system and each other.

One of the first decisions that need to be made is how the camera and projector are going to be set up. Based on this decision, the second part to consider is the system design which describes the art style and events that influence the projection. The goal of this project is to generate art that people can control based on movements. Therefore, the installation needs to be able to attract a crowd in order for it to be successful. Next, the attracted crowd needs to be entertained and motivated to stay. Therefore, the design should encourage participants to explore, work together and interact with each other and the system.

When the system design is complete, the software is the next part to consider. Image processing techniques have to be implemented to apply to the video stream and detect where people are. Based on this information, graphics have to be generated that can be projected onto the scene. To make the system flexible, a calibration stage needs to be added for optimization and adaptation to every location. These parts have to be combined and tested.

When these parts are implemented and well tested, the system should be ready to be installed in a public space. A full scale installation can be set up on location.

3

Research

This chapter covers the research that we have done over the first two weeks regarding the different topics needed to complete this project. The first section contains the results of our interaction design research, which is then applied in section 2, where examples of interactive systems and our own design ideas are given. In section 3, the hardware choices and calibration techniques are covered. Then, in section 4, we cover the image processing software and techniques, and in section 5, we cover computer graphics software to project the designed art.

3.1. Interaction design

For an interactive system to be effective, it is necessary that a crowd actually wants to participate. Without people interacting with the system, it is useless. Therefore, we first of all need to figure out how to attract a crowd. The system needs to catch people's attention. Once it caught some attention, people have to start participating in the interactive art. Because the goal of this project is not only to make people interact with the system but also with each other, multiple people need to be participating. Therefore, we will take a look at possibilities and techniques to encourage participation.

3.1.1. Crowd Attraction

Before people start to engage with the system, it has to draw them in and attract a crowd. The first step is to make people actually walk up to the projection before the interaction happens. Brignull and Rogers [12] researched this subject using an interactive screen where people could post opinions on a displayed subject. They identified three spaces of activity regarding public interaction with the installation:

- Peripheral awareness: People do not notice the installation or merely take a look at it. They know nothing about it and no further engagement is involved.
- Focal awareness: People engage in socializing activities associated with the installation meaning that they are talking about, gesturing to or watching the system, possibly while it is being used by others.
- Direct interaction: People interact with the system.

Our goal is to convince people to participate, meaning that they have to transition from the peripheral activity space to the direct interaction activity space. From focal awareness to direct interaction will be discussed in 'crowd participation'. Crowd attraction is about the transition from peripheral to focal awareness. Taking some examples of interactive systems

and their corresponding conclusions into account (e.g. [12], [25] and [10]), we can conduct a design plan for crowd attraction specifically for our system.

There are two scenario's we need to consider: an empty field and unknowing by-passers. When the field is completely empty, the projection needs to show something that makes people curious enough to go investigate it. When the field is not empty because of people walking into the projection area, not knowing that it is an interactive system, something has to catch their eye to point out the interactivity. This action needs to be clear enough to be noticed by the walking person himself causing him, the unknowing by-passer, to stop and investigate or by bystanders to encourage them to walk op to the projection. These luring actions need to evoke positive conceptions to be able to entice people to move towards the projection [12]. This will most likely not convince anyone to step onto the scene yet which means that the software has to detect bystanders and show something that clarifies the interactive purpose and makes them step into the projection area. From there, as soon as people noticed the projection and engaged by entering the projection scene, the art must change based on the interaction. Unfortunately, because crowd attraction focuses only on creating awareness for the installation, this is not enough to make people participate. Now that the system gained some attention, we need to make sure that the attracted crowd actually participates.

3.1.2. Crowd Participation

According to Maynes-Aminzade et al. [17], people lose interest within 30 seconds if the interactive experience is not entertaining enough. This means that when a few people appear on scene, the interaction has to start right away and make everyone want to participate. Furthermore, as soon as there appears to be some participation and the interaction with art has began, participants need to remain entertained during the whole experience to prevent them from discontinuing. Churchill et al. [15] confirms this as they found out that users need constant encouragement to keep interacting with these kinds of public interactive systems. Maynes-Aminzade et al. [17] defined seven design principles to optimize audience participation in interactive systems out of which four of those apply to our project as well:

Focus the design on the activity and not the technology Even though interactive systems like this are not very common and people are easily impressed by the technology, it is very important to shift their focus to the activity to ensure participation.

Make the control mechanisms obvious Intuitively this means that mechanisms have to be easy to understand and execute. In our project, because people can arrive and start participating at any time during the experience, it is not possible to show everyone a tutorial to explain the controls and therefore they need to be easily derivable from the combination of movement and projection. On the other hand, making the control mechanisms obvious also means showing people that they actually are in control. This can be done by optimizing the controls to make projections accurately follow movements.

Vary the pacing of the activity This principle focuses more on interaction design instead of system design in general. Maynes-Aminzade et al. claim that interaction with high intensity work best when combined with periods of relaxation. If our system contains multiple types of arts or different phases, they should be combined with short pauses to allow the audience to cheer and prepare for the next part.

Ramp up the difficulty of the activity As mentioned before, in our case a tutorial is not an eligible option to explain mechanisms. A better way is to start off with a fairly simple interactions to make the crowd discover the controls and offer more advanced controls for the additional effects.

Now we can deploy these principles in the design of our system as described above. These techniques should optimize crowd attraction and crowd participation and contribute to an entertaining interactive experience for everyone.

3.2. Crowd Art

Next, we will show some examples of actual interactive systems and finish with two design ideas for our project.

3.2.1. Examples

Public interactive systems using real time crowd tracking are currently not widespread. Most are temporary, non public, small crowd installments, or not tracked by computers. We list some of these examples below, which fed our choice for interaction design.

Spotlight

ACCESS lets you track anonymous individuals in public places, by pursuing them with a robotic spotlight and acoustic beam system [38]. The spotlight is moved by a human watching a live camera feed using a joystick. However two beware-statements are given by Marie Sester: Some individuals may not like being monitored, some individuals may love the attention [39].

Interactive Art Wall

This installation wall uses a Kinect [43] to let shoppers take control of the iconic Stachus gate and thereby bring the ancient medieval structure to life by jumping, dancing, waving and gyrating.

Gymnastics Motion Art

A series of animations based on the motion of athletes. Even though it is not real time and created by human hands, it is derived from the original footage to create motion art from movement [26].

Red Nose Game

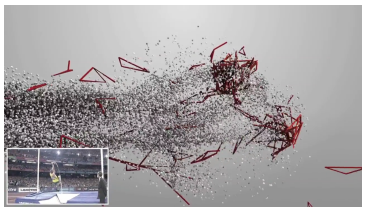
The Red Nose Game [30] is a collaborative game played on the BBC Big Screens. The game begins with small red 'blobs' (like clown noses), placed randomly across the screen. The camera is put above the space in front of the Big Screen where the players are located. The goal of the game is to push the virtual blobs together into a single big blob. Image processing techniques are used to determine people movement using the live feed of the camera.

Snowball

Electrabel's TV commercial eindejaarscampagne 2013 [19] shows two people interacting with a LED display showing winter effects like snow, snowball fight and creating a big snowball. The wall is a vertical interaction display. Effects are probably not interactive, due to it only being a commercial.

(a) ACCESS Spotlight

(b) Interactive art wall using Kinect



(c) Gymnastics motion art

(d) The Red Nose Game

(e) Electrabel TV spot eindejaarscampagne 2013

3.2.2. Designs

Below we designed two interactive art ideas that follow the guidelines specified earlier to increase participation. For all arts hold that the scene should never be in a static state and for every action a visible rewarding effect should be included.

Fluid

Fluid is a kind of art that acts like real life fluid which changes color, speed and location based on people's movement through the projection. This can also include changes on random inserted places to keep the scene dynamic and attractive. An example of what the projection might look like can be seen in Figure 3.2. This idea follows the guideline to create curiosity in case of an empty projected field to grab the attention of passer-by crowd and attract them to participate. This also follows the guideline to create curiosity when one or multiple people walking through the projected art: fluid will move based on the movement of the people. Each person gets assigned a primary color which they spread by moving. To create a longer participation, objects can be included, such as a ball, to be guided through the fluid.

Figure 3.2: An experimental fluid simulation

Light Trails

Light Trails is a kind of art which draws ever moving glowing dots leaving a light trail that swarm around each person on the scene. Two people close to each other exchange these light trails. When a person leaves the scene, his light trails move to a person who is still in the scene to encourage engagement. When there is no one on the scene, the light trails pick spots to circle around to activate interest of passer-by crowd and encourage them to interact with the light trails. Each person entering the scene starts with a unique color of light trails. This idea was inspired by the iPhone app Spawn Glow [29] visible in Figure 3.3.

Figure 3.3: iPhone app Spawn Glow [29]

3.3. Hardware

To fulfill the requirement to have a setup of projective art and use tracking of movement, the following choices have been made.

3.3.1. Setup

In this subsection we will select the most likely setup options to a feasible setup for creating the procedural crowd art.

Player Perception

Multiple solutions exist for the player to be perceiving feedback: audio, visuals, vibrations, smell and taste. Smell and taste are however not (yet) far developed for this purpose. For a public space, vibrations and audio feedback are very intrusive, so only visual feedback remains. Multiple solutions for visual perception exist: lasers, LED walls or floors and projectors. We chose an off-the-shelf and a fast-to-implement solution, the projector, to concentrate on the actual interactive art in software. The projector should have a resolution of about 720p to prevent a pixel perception of the player. Additional requirements are added when using a projector:

- A location with a high ceiling with a clear space between floor and ceiling that allows the projector to display the scene unimpeded in the required size.
- A location with a low light setting and a minimum of daylight.
- The ability to hang the projector on a high place.
- A projector with approximately 5000 ANSI lumen, depending on the of amount external light [16] [8].

Tracking Players' Locations

There are multiple ways to track the players: high participation threshold player-added-technology such as smartphone apps or other external devices for measuring, but also low participation threshold technology such as radar, laser, infrared, sound or camera. To achieve the highest participation rate possible and use the cheapest off-the-shelf solution available, a regular video camera will be our choice. The resolution to be able to capture individuals from a high distance is required to be about 720p.

If projection subtraction, described in subsection 3.4.2, is too time consuming in terms of development and therefore not feasible, an alternative is to use an infrared camera. A drawback of the infrared information is that a surface heats up when a person sits or lays down for a while. When the person leaves the spot, it leaves a temporal heat signature. This may result in the detection of two people instead of just one. A method would be required to be implemented to eliminate this infrared trail stamp.

Processing Device

To process the input and transform it to the output art, we will require a device which can handle camera input and projector output. A common computer, laptop or embedded device with a regular CPU, USB and VGA or HDMI will be enough, preferable including a Ubuntu operating system, but probably a Windows system for the installation setup. Further it should be able to process the visual input and output fast enough to prevent participants from experiencing latency which will cause them to leave quickly.

3.3.2. Calibration

The setup of projector and camera requires the located participants to be mapped to projected dots on the ground. Arising challenges are: non orthogonal projector and ground and ground-camera angles for x, y and z parameters, non parallel projector-camera setup and maybe even including lens corrections.

Camera calibration

Camera calibration can be done using a checkerboard plane on the floor [31]. The camera distortion can be calculated using the pinhole camera model extrinsic and intrinsic properties. This distortion can be used to change the input image to create a non-distorted square image.

Projector calibration

To calibrate a projector, the approach used for camera calibration can be applied the reversed way. This method also requires a projected checkerboard pattern on the floor. It also requires a calibrated camera which captures the complete projected image. The captured image is used to find the inverted pinhole camera model of the projector to be able to project an image which is non-distorted and square.

Self-calibration

He Zhoa [44] created an application which retrieves the camera and projector calibrations from a series of images using the non-calibrated camera and projector. The implementation is for e.g. Windows and OSX and might require a port to Ubuntu for development. This self-calibration will speed up our setup time.

Detected object to projector coordinate mapping

Daniel Moreno and Gabriel Taubin [28] show the mapping between the captured image and the image to be projected. This can be used to map detected objects to coordinates onto the projection image for processing location and movement into the art to make it interactive on the correct positions.

3.4. Image Processing

To detect participants and their movement, image processing techniques need to be used. In this section, we will cover the software aspect of image processing, and the various methods that we have tried or learned about with regards to solving the problems that come with finding and tracking participants and their movements.

3.4.1. Software

The choices made in the software aspect concern the library that we are going to use, and in which language we are going to employ said library.

Libraries

We prioritized selecting a good, well-documented library over picking a specific language; the quality of the library is a really significant factor in our development speed, whereas an unknown language is often easy to adapt to. We have found several options.

OpenCV is the industry beast used by big companies like Google and Microsoft [4]. This is our first and most straightforward option. There is a lot of documentation on this library and there is an active community. These are very important factors when trying to solve difficult problems using this library. *OpenCV* is slightly low-level, and is therefore a little hard to get into. Conversions between different kinds of images and interpretation of method results can often prove to be difficult and provide significant overhead. However, when used right, *OpenCV* can provide significant performance.

CCV is the next found option [2]. This was quickly dismissed, however, due to its severe lack of documentation and an active community.

SimpleCV is one option that is considered due to its simplicity [6]. This is a Python library that is basically a wrapper around OpenCV and some other image processing libraries or algorithms, to make them easier to work with. This can be seen as the more high-level version of OpenCV. It allows for fast prototyping, but requires to implement C-bindings for Python for self created high performance algorithms [32].

Because we need to be able to process the input of the camera real-time, performance is a high priority for us. Therefore, OpenCV seems like the most sensible option. It allows for maximum control and simplicity to develop the whole system in the same language. SimpleCV may be used to quickly test different kinds of approaches on a higher level.

Language

Since we picked OpenCV for its low-levelness, it seems sensible to pick a language that follows that trend. C++ seems the most straightforward choice, simply because C++ is the only language that OpenCV works in natively. Working in Python or Java would significantly reduce performance or requires creating bindings [32].

3.4.2. Methods

In this subsection, we will cover the different ways to solve a multitude of problems that are posed in the process of spotting and tracking people and their movements, along with the considerations that go into picking the right approach.

Background subtraction

There are many ways to manage background subtraction. We tried out a few of the lightest ones in OpenCV.

(a) Two men (b) The background (c) The difference

Figure 3.4: In (c) you see the result of the operation (a) - (b).

In the first approach, one frame is simply selected as a background, as seen in Figure 3.4b. Then applying a simple minus operation:

$$\textbf{diff} = \textbf{image} - \textbf{background} \tag{3.1}$$

And after that applying a simple threshold for every color channel i:

$$\text{thresh}_i = \begin{cases} 255 & \text{if } \text{diff}_i \geq 50 \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Figure 3.5: The result from thresholding the difference in Figure 3.4c

(a) Two women (b) The difference (c) After threshold

Figure 3.6: A case where this manner of background subtraction does not work.

By applying this threshold we get the result you see in Figure 3.5. The men can be distinguished fairly well. However, there is also some background noise that is detected as foreground, which might pose a problem. There is another issue, though, as seen in Figure 3.6. The women have colors very similar to the background, and as a result they fail to get past the threshold. Because of that, they are not recognized. On top of that, this method is not at all resistant to changes in the luminosity of the area filmed; the sun might start

shining brighter, causing more daylight to show up. In such a case, all of the image would be classified as foreground.

(a) Two women (b) KNN result

Figure 3.7: The result of the application of the K-Nearest Neighbor background subtraction algorithm.

Taking all these issues into account, it seems like we need a more complex approach. When applying the K-nearest neighbor (KNN) algorithm as covered by Zivkovic and van der Heijden [45], we see the results in Figure 3.7. In the experiment the KNN algorithm shows a better result. Additionally, since this algorithm adapts every frame, it can adapt to changes in lighting just as in this experiment.

Should we want to experiment with other algorithms, M. Piccardi [35] has given a clear overview of what a number of options are. The algorithms by Zivkovic and van der Heijden have the advantage of already being implemented in OpenCV.

Shadow elimination

In order to detect objects successfully, we need to make sure that shadows are not detected as 'foreground' by the background subtraction. Prati et al. [36] have given an overview of numerous algorithms to spot shadow in a frame. The KNN background subtraction in OpenCV employs one of these algorithms to find shadows, and gives them a grey color. This works very well, as can be seen in Figure 3.8.

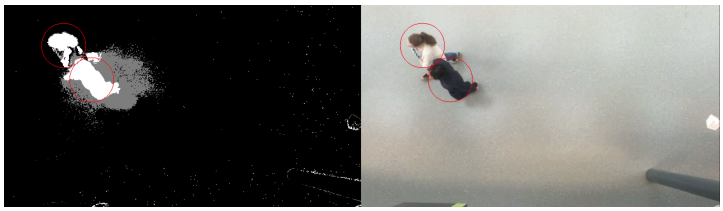


Figure 3.8: Detection of a shadow. The black values are background, the white values are foreground, and the gray values are shadow.

Object detection

The first and simplest step in object detection is simply finding areas where a lot of pixels are classified as foreground. This is called blob detection. This yields the results seen in Figure 3.9. If two people are far enough away from each other, blob detection works just fine. However, if they get too close together, they will be seen as a single blob and therefore a single person. An example of this can be seen in Figure 3.10. Of course, the higher resolution of this second case may also play a role. Putting a camera as high on top as possible will reduce those errors.

Figure 3.9: Labeling of the people in a scene using blob detection. The red circles denote a part that was labeled as a person.

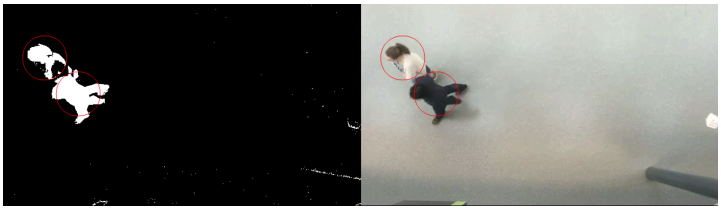


Figure 3.10: An example of side-by-side detection working with a camera from high up.

A different option would be to disregard the background subtraction altogether and use a Histogram of Gradients (HOG) approach instead. However, this is a very slow method, and also does not prove to be very accurate, as can be seen in Figure 3.11. We are going to look into the use of background subtraction to find areas of interest, and then use a HOG detector only on these areas.

Figure 3.11: The result of using HOG detection.

Object tracking

When we have found in a frame where the people are, a new challenge arises: we need to know which object represents which person. This is useful knowledge when we want to create person-specific effects. For example, we might want to give each person a different color to draw with.

The simplest approach to do such a thing would be to simply say that a person in frame i corresponds to the person in frame $i-1$ with the smallest euclidean distance to the person. This could pose a problem when people get too close together, however; they might 'switch

places', so to say. Keeping track of the color histogram of a person could be a good solution for this problem.

Motion detection

When people are on the field, we might want to do something special with their specific movements. For example, when they thrust their arm forward, or when they spin around, something should happen. However, this is a very difficult task. An appropriate way to approach this task is by using Motion History Images (MHI) and Motion Energy Images (MEI) to distinguish certain motions. [18]. Using both of these types of images allows for a great distinction between different kinds of motions, since certain different motions can be similar in either MEI or MHI, but generally not in both [18]. A drawback from this approach is that these images need to be compared to a set of images that are seen as 'model images'. Therefore, this approach requires training.

Calibration and projection subtraction

When projecting on the plane that is also filmed by the camera, the camera will also perceive the projection. Somehow, we need to get rid of this projection from every frame before applying any of the other methods mentioned. In subsection 3.3.2, we cover the calibration of the camera and the projector. When calibrated, the location of the projection in the image perceived by the camera is known. Using that data and the actual projection generated by the graphics part of the software, we can subtract that projection from the perceived image to a certain degree. What exactly that degree is and what other issues we may face, we will see when we set up a testing environment for this situation.

3.5. Computer Graphics

In this section we will first choose a graphics library. After that, we validate our choice for this library by creating an experimental projection image using this library.

Graphics library

A lot of graphical libraries are available. We have selected a list of graphics libraries that run on our development and setup platforms: Ubuntu, Mac OS X and Windows, and are available for the language selected for image processing to limit the development scope to a single programming language: C++. Another requirement is that the libraries must be available for free. Below we listed some popular libraries that follow these requirements.

FreeGLUT is a free-software/open-source alternative to the OpenGL Utility Toolkit (GLUT) library. It is a cross-platform library that is written in C [1].

Simple and Fast Multimedia Library (SFML) is a cross-platform software development library written in C++, designed to provide a simple application programming interface (API) to various multimedia components in computers. It also provides a graphics module for simple hardware acceleration of 2D computer graphics which includes text rendering [5].

Simple DirectMedia Layer (SDL) is a cross-platform development library designed to provide low level access to i.e. graphics hardware via OpenGL and Direct3D. SDL officially supports Ubuntu, Mac OS X and Windows. SDL is written in C, works native with C++ [3].

For development using frequent high level utilities and developing in the C++ way, we will choose SFML for our art.

SFML Experiment

To validate our choice for the graphics library, we created an experimental projection image with the SFML: fluid. A beautiful effect would be to simulate fluid flowing over the projected plane. This would allow for the people to influence this fluid with their movement and get a feeling of immersion as they fill the scene with color. Such a fluid effect requires some kind of simulation. Simulating many 'fluid particles' can be very computationally heavy and is not fit for real-time use [41]. Instead, we would like to solve the differential equations that describe the physical state of fluids. Jos Stam [41] proposes a quick Navier-Stokes equation solver that is geared to speed over accuracy, so that it is fit for real-time use. Implementing this solver and mapping the results to color values has yielded the results that can be seen in Figure 3.12.

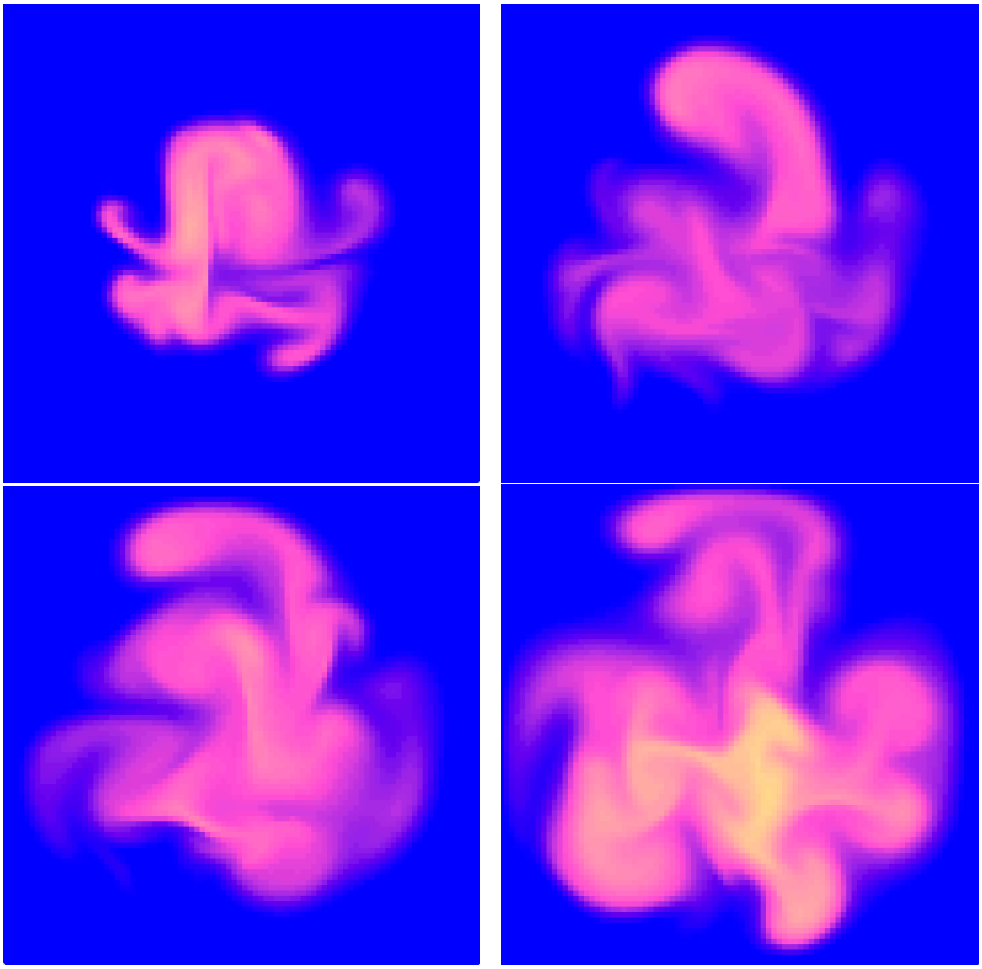


Figure 3.12: A fluid simulation using the Navier-Stokes solver proposed by Jos Stam [41].

Conclusion

Based on the goal of the project, to let passerby-crowd generate, interact with and control projected art, and based on the requirements, we investigated interaction design for crowd attraction and participation. We looked up examples of crowd art which do also achieve the same project goal. We came up with Fluid and Light Trails as designs for our project art and continued with choosing a regular camera, projector and general computer as hardware to fulfill our requirements and looked at the calibration of the camera and projector. After that we explained how to achieve this with image processing using the OpenCV library and methods such as background subtraction to accurately detect objects and motion. And we finished by explaining our choice for SFML as graphics library to display the art.

4

Development process

This chapter will describe the chosen development methods, how the project is organized using these methods and the general planning of the project.

4.1. Development methods

This section will explain the used development methods: upfront planning, time boxing, separation of responsibilities, continuous integration, pair programming, pull requests and testing.

4.1.1. Upfront planning

For our project we have created a general planning upfront. This suits the project, since the time scope is constrained and small. We have applied those constraints on the project planning as well. As a starting point we used the already defined planning: the first two weeks are spent doing research and next six weeks are spent on the implementation. We planned for adaptation of the planning in the first two weeks of the implementation phase.

4.1.2. Time boxing

To increase control over our development process we have used time boxing [27] for features and fixes in time box sizes of a week and of a day. Each week in the implementation phase we started with a meeting on what we were planning to achieve that week. In the second half of that week we evaluated the progress and if required we changed our approach to the problem or extended the time required for the specific part. Each day we started with a meeting on what we were planning to work on that day. In the second half of the day we evaluated the progress on each part.

4.1.3. Separate responsibilities

At the beginning of the project, each team member chose a specific part of the system to be responsible for. The team member did the researching, implementing and testing of that part and processed feedback other team members gave about their implementation. The three parts for our project are the light trails scene (section 6.4), people detection (section 6.3) and the connecting component, which in turn consists of calibration (section 6.2), projection elimination (subsection 6.3.4) and the combining architecture (section 6.1).

4.1.4. Continuous integration

[23] Due to separation into three parts, it is required that the parts are joined together into a single system. We choose to do this as soon as possible on every finished change in a part of the system. This way, we made sure to maximize the mergeability, because deviation from the main branch is as small as possible.

4.1.5. Pair programming

Because changes in the implementation of the interface, the configuration or utilities may conflict with other parts of the system, we used pair programming [7] for integration of these changes into the system. For each part of the system that is changed, the responsible team member joins the merging developer to use both of their specific knowledge on each part to find the best way to merge the changes into the main system. This includes taking care of the deviation from the main branch.

4.1.6. Code standards

For code standards we have used Object Oriented Programming (OOP) [37], Software Improvement Group (SIG) [22] maintainability check, the c++ language, which is defined strict in itself, and standard compiler warnings as errors. OOP has required us to think in a model driven way instead of imperatively, since the major challenge is to create the right model. The SIG check has kept the code maintainable for the duration of the whole project including changes after the time scope had finished. Standard compiler warnings as errors prevented common pitfalls and created a compiler (not operating system) independent program.

4.1.7. Pull requests

Before a feature was merged on the main branch of the project, a pull request [24] had to be made. This pull request was then reviewed by a team member and merged into the main branch if the code complied with the code standards and appeared to be functionally correct. Otherwise the discussed changes had to be made and again a pull request needed to be made.

4.1.8. System testing

We tested each part separately and tested the system as a whole [13]. Every individual part was tested using specific methods chosen by the person responsible for that part. Complete system tests were done by integrating all parts and running them on either a laptop or a setup with camera and projector.

The testing methods and their results are more extensively covered in chapter 7.

4.2. Organization

In this section we lay out the tools and services we used for our project to control and automate our project. We used Trello for progress tracking, Git for code versioning, GitHub as a central resource for Git and for pull requests and SIG for maintainability control.

4.2.1. Progress tracking

For tracking the progress of the project we used the digital card board service Trello [42]. We used three card categories to organize our development process: To do's, Doing's and Done's. This gives an overview of what needs to be done, what is currently being done and what is finished. These cards were used to track features, bugs, refactorings and communication. Cards associated with a deadline had a due date for a clear overview of whether we were running on schedule.

4.2.2. Code versioning

To control and automate continuous integration as explained in section 4.1.4, we used Git [20]. Git allows to store the whole code base including every change in the history of the project in a single repository and it helps merging code of different versions. For experiments we also used git to create branches and merge a few features together to do a quick partial system test or an experimental demo.

GitHub [21] was used as a freely and privately available resource to be our central server for pushing and fetching our code. This made sure we have a central storage and authority for our code branches.

4.2.3. Maintainability control

At the end of the fourth implementation week, we made use of the required submission of our code to the Software Improvement Group (SIG) [22] for review by experts on the subject of maintainable software development. Their feedback on our code was evaluated and changes were applied where needed. A final maintainability check was submitted on the final code which included those changes to check whether the code maintainability level changed compared to the previous check. Due to the limited duration of the project, the result of that check will not result in any changes for this project period, but will be of educational benefit for future projects.

4.3. General planning

First we started with a general planning for the whole project. The project was required to be split in two phases: the research phase during the first two weeks and the implementation phase during the other weeks.

During the research phase we focused on people detection methods, rendering scene libraries and the mapping between camera and projector.

In the first week of the implementation, the general upfront design of the project was created and thereby the interface between the people extractor and the scene. In the same week the scene system design was created upfront and discussed with the client. This would define the major scope of the project which is part of the project description to create this scene system design (Chapter 5). Minor changes to both the general system design and the scene system design were allowed throughout the whole project based on specific unseen needs for features.

The focus of the next weeks was implementing the different part separately. In the second implementation week the individual system parts were brought together for a first integration and tested if they worked together properly. The next weeks this was repeated in a shorter cycle. All parts were brought together and tested multiple times a week.

Together with that cycle, the following weeks were about testing in the real world. In the third and fourth weeks the focus was on a small real test setup, to actually use real-time video for an the interactive scene. The fifth and sixth week were all about testing on the exposition location. The seventh week was a gift due to the presentation date of the project being scheduled one week later than initially planned. This week was used for further testing on the exposition location and start deploying the actual installation. The last weeks were used for creating a report and a presentation of the project.

5

System Design

This chapter covers the design of the system. The system is what people see and experience when they pass by or participate. The design describes everything that happens on the scene.

5.1. General

We start off with some general aspects of an interactive system using a camera and a projector.

5.1.1. Definitions

Camera Area on the surface which is captured by the camera.

Scene Area on the surface which the projector projects the art upon.

Bystander zone Area on the surface 0 - 1 meter distance from the scene.

Bystander Person who stand still on camera and is in the bystander zone.

Passer-by Person who walks through on camera, not entering the scene and not standing still in the bystander zone.

Participant Person who is in the scene.

5.1.2. Vision

The goal of the system is to let passerby-crowd generate, interact with and control displayed projective art (see chapter 2). From this follows that moving is preferred above standing still, being a bystander over a passer-by, being a participant above a bystander and person - person via system interaction above person - system interaction.

5.2. Light Trails

Now that the general aspects have been explained, a more in-depth description will provide all the details specific to the system design idea we developed.

5.2.1. Definitions

Light trail A trail of light with a specific hue, 100% saturation and 50% lightness, specific speed and specific direction.

Light source A point in the scene that sends out light trails at angles, speeds, and hue within a certain range.

Edge An edge of the scene. It reflects light trails as though it is a mirror.

Gravity point A point on the scene that attracts light trails within a certain hue range, and is meant to cause them to orbit around itself.

Anti-gravity point A point on the scene that repels light trails within a certain hue range.

Color hole A gravity point that sucks up light trails within a specific hue range. When a participant is standing on top of the hole, all sucked up light trails shoot out of it and return to the person who is standing in the middle of the hole.

Mixing The process wherein both participant's gravity points change hue to their combined average. The light trails within one meter change hue accordingly. The result will be that the light trails will be exchanged and the participant's orbiting light trails hue average should be the same for both participants.

5.2.2. Initial state

- A light source in each corner, corresponding to 4 different distinguishable hue ranges. Each light source is also a weak gravity point for its hue range, and a very strong anti-gravity point for the complement of its range.
- 2 types of 5 seconds time slots, time slots change alternating.
  - No gravity points: random moving light trails across the screen following their last trajectory.
  - Add gravity point.

5.2.3. Time-based events

- Every 30 seconds, a color hole appears for a random hue range at a random location if there is not already a color hole (since the last one may not yet have been removed).

5.2.4. People-based events

Empty camera Same as initial state.

Person enters camera Nothing changes.

Person enters bystander zone Randomly assign a specific hue range to person. The color is one of the colors of the light sources of the corner. Shortly place a gravity point of that specific color at the point in the scene closest to the person.

Bystander stands still Assign specific hue range to person. In alternating time slots do create a gravity point for the assigned hue range at the point in the scene closest to the bystander.

Participant enters scene A gravity point of the already assigned hue range is created on spot of person, along with an anti-gravity point corresponding to the exact opposite of the assigned hue range.

Participant enters empty scene All light trails of the person assigned hue range go directly to the participant. All other trails directly return to their sources, but reflect on the sides of the scene.

Participant enters non-empty scene New light trails from the corresponding source move towards the participant.

Participant stands still on scene Light trails orbit around the participant's gravity point. The longer the participant stands still the weaker its gravity point will become, so that light trails orbit further away from the participant and maybe even get picked up by other participants.

Participant moves on scene Light trails follow the participant's gravity point. Gravity gets stronger again if the participant was standing still and his gravity was weakened.

Two participants are within two meter of each other :

Participant enters radius Light trails from participants will start exchanging and mixing if the difference between their hue ranges is above a certain threshold. A countdown of 5 seconds is set after which mixing is complete. If countdown is 0, mixing is complete, both participants receive the newly created color. At this moment the light trails fly away from the gravity points for a short moment and return to create a short explosion effect.

Participants get closer each-other The time before mixing completeness is linearly reduced, this is will encourage participants to get closer and interact.

Participants distance themselves The time before mixing completion gets longer to a maximum of five seconds. This encourages participants to stay close together and interact.

Both participants move without distance changing Nothing changes.

Participant exits radius If the mixing countdown is not 0, hue of both players will change to initial hue before mixing and the connection breaks. Otherwise the connection simply breaks, the players keep the mixed color.

Participant exits scene The participant's gravity point is removed. The light trails are shot away upon this removal and will end up at the other participants.

Participant exits bystander zone Participant's assigned hue is removed.

Participant exits camera Nothing changes.

Participant is within 1 meter of light source If the color of the participant is different from the source, all his light trails fly away and he receives new ones from the source. If the color is the same, the source and player exchange some light trails.

Participant stands on color hole The color hole explodes, sending out all the light trails it sucked up at great speed. It then ceases to exist.

6

Implementation

This chapter covers the general software design and the important implementation parts of calibration, people detection and the light trail scene.

6.1. Software Design

I. M.O.V.E. consists of two major applications: the I. M.O.V.E. application itself and the calibration application. The I. M.O.V.E. application is split into two systems: people detection and the scene.

The people detection reads the calibration configuration, which is created by the calibration application and uses this to read frames from the configured camera. After detecting people from the camera frame, the people detection manager pushes the set of detected people in the queue for the scene to be read. The scene pops the detected people from the queue and uses these to show the described effects on the scene. The interaction between different parts can be viewed in figure 6.1.

The people detection and scene both run in their own process and use shared memory for communication. Due to the long computing time requirement of people detection for each camera frame, the scene and the people detection both need to run on its own CPU core to keep a high enough frames per second for the scene for real people not to notice stuttering. Due to a limitation of OpenCV [14] and SFML on OS X [40], both systems need to run in the main thread which results in a separate process for each system. This adds a communication challenge. For fast communication we use a shared memory between the systems as inter process communication. The Boost library [11] is used to ease the development of shared memory communication. To further speed up the communication the scene frame copying is done in a separate thread in the scene process and in the people detection process.

6.2. Calibration

Several parts of the system that differ for every camera - projector setup are required to be measured and set as input. This influences the projection, real size per pixel, projector and camera and the projection subtraction settings.

6.2.1. Projection

The coordinates of extracted people on the camera frame need to be mapped to scene coordinates. This mapping is done using the perspective transformation offered by OpenCV [34]. The input values are the coordinates on the camera frame. The output values are the coordinates of the scene frame corners. The last parameter is the perspective transformation matrix.

OpenCV offers a method which creates this perspective transformation matrix [33] from four coordinates on the first plane, in our case the camera frame, and four coordinates on

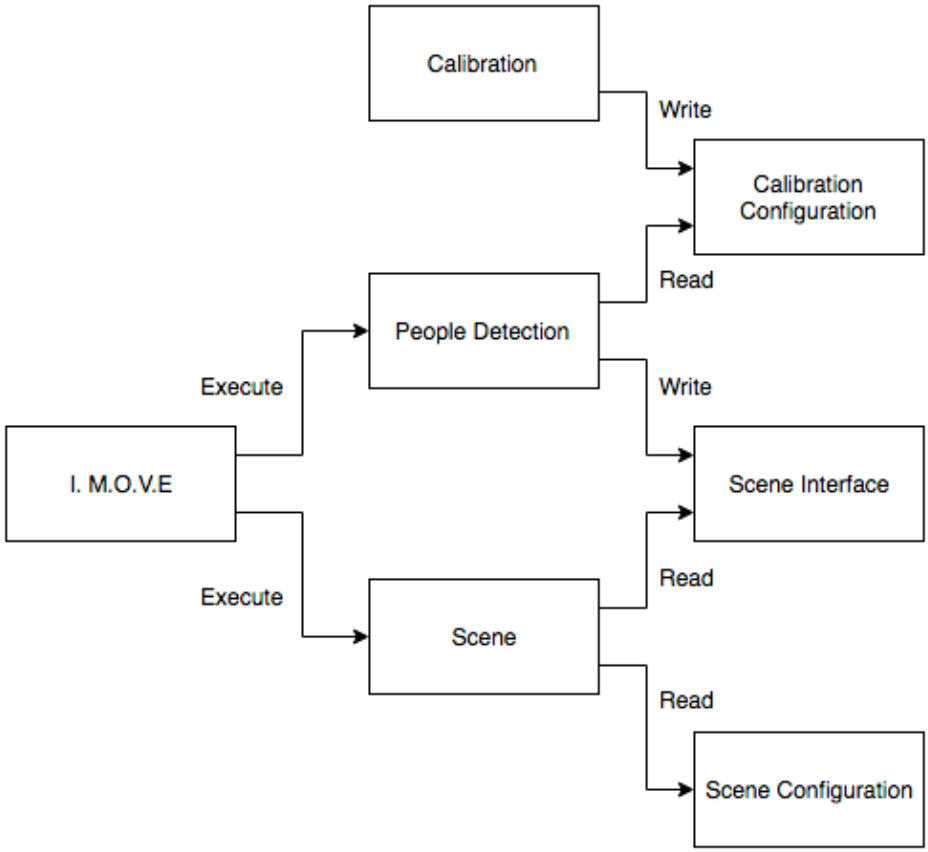


Figure 6.1: The high-level design of the software.

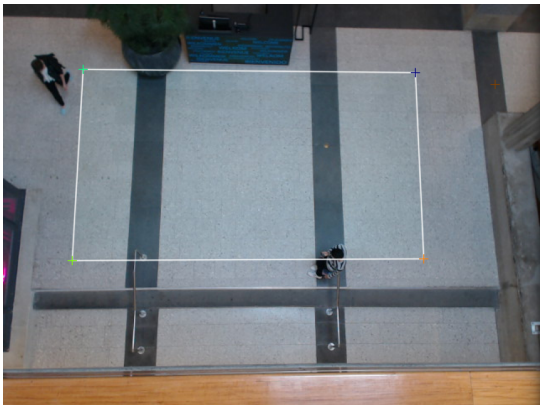
the second plane, in our case the scene frame. The four coordinates on the camera frame are the set coordinates using the mouse which point to the four corners of the projection on the camera frame. The four coordinates of the scene frame are the four corners of the rectangle scene.

In every setup, the camera and projector have a different distance between each other. They also have a different distance towards the floor as well as a different angle. This means that for every setup the perspective transformation matrix needs to be calibrated. The calibration is done by indicating the four corners of the projection field of the camera frame using the mouse pointer. The projection calibration can be seen in figure 6.2.

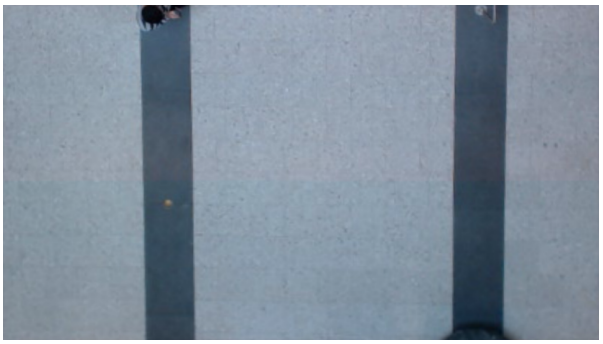
The researched method in subsection 3.3.2 of using a chessboard for calibration was not very accurate and simple. The lens of both the projector and camera contain internal calibration, so no noticeable disturbance can be found. The perspective transformation is more intuitive, accurate and simpler for this project and is therefore the best option for this project.

6.2.2. Meter - pixel map

Both the blob detector in subsection 6.3.2 and the scene in subsection 6.4 require to know the size of a meter in pixels. The blob detector uses this to be able to classify the right size of blobs as a person and the scene uses this as base unit for parameters such as the size of a scene element. To be able to support this, the meter - pixel mapping is calibrated on the camera frame as a distance between two points. These two points are set using the mouse pointer with a measured distance of 1 meter between these points. An actual known meter object or surface is required to be put on the floor on the camera frame in the calibration phase. The meter - pixel calibration can be seen in image 6.3 below.



(a) Projection boundaries set by mouse pointer using a white screen on the projector.



(b) Projection frame based on crop of camera frame. Here: Approximately the scene top left captured on camera frame bottom right.

Figure 6.2: Calibrating the projection boundary coordinates on the camera frame using a mouse pointer.

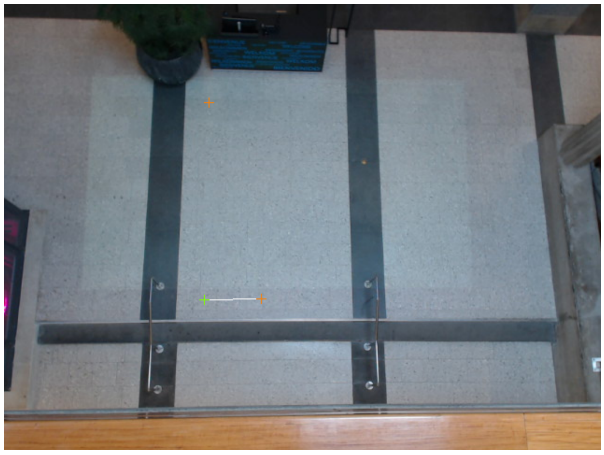


Figure 6.3: Calibrating the meter on a camera frame using two set points by a mouse pointer measured on a known length.

6.2.3. Projector and camera settings

For the perspective transformation and real size in pixels, the resolution of the camera is required. For the perspective transformation, the projection resolution is required as well, or at least its ratio. The camera settings are read using OpenCV, and the projector settings are set by the operation system. The projector/scene resolution must then be set in the calibration configuration.

Besides the resolution, the next thing is to control the focus of the camera and the auto balancing of colors. A refocus of the camera can cause some disturbance for people detection and requires it to reset itself. After the camera is focused, the auto focus is turned off. Auto balance of colors is currently assumed to be done by the camera. This way, light changes due to clouds or sunlight and the day-night cycle are automatically solved. This holds, assuming the light changes are not fast.

6.2.4. Projection subtraction settings

For the calibration of the projection subtraction in section 6.3.4, a few parameters need to be calibrated, because they differ on each installation. The parameter of the intensity of the subtraction, which is based on the environment light, has to be configured. The synchronization between rendered scene frames and captured camera projection frames has to be configured as well, so that the most accurate scene frame is subtracted from the camera frame. Finally, the optimal balance between the amount of frames and the resolution of the

frames has to be configured. This depends on the resolution of the scene, the minimum fps of the scene, the speed of the camera and the speed of the machine.

6.3. People detection

The second part of the implementation is the analysis of each frame received from the camera and detection of where people are in the scene. First, the background needs to be removed so only the people remain in the frame. Next, blob detection can identify people's locations and a small correction of those locations based on perspective gives us the location of their feet. Finally, those detected locations have to be matched with people from previous frames to know where they moved to, new people have to be created for newly detected locations and people who left the scene have to be removed.

6.3.1. Structure

The image processing part is split up into four parts:

- PeopleDetector: Analyses each frame and detects locations where people would be.
- PeopleIdentifier: Matches new locations to the right people, creates new people for unmatched locations and deletes people who left the scene.
- Projection elimination: Removes the scene projection from the camera frame to prevent disturbance.
- PeopleExtractor: Prepares the frame for detection (resizing and converting to gray scale), passes the frame to the detector and passes the detected locations to the identifier.

6.3.2. Detector

Background Subtraction

The detector receives a frame to which it applies the K nearest neighbors background subtraction from OpenCV. As mentioned in the research, this method works best on our test videos to remove backgrounds and detect shadows. It also adapts to background changes easily. The result is a gray scale image containing only three intensities: zero intensity for areas detected as background, full intensity for areas detected as foreground and half intensity for shadows. Thresholding the image to keep only the full intensity parts leaves us with a frame with white blobs which should represent people. Using OpenCV's blob detection we can find the center of these blobs and the location of the people on scene. The result of this can be seen in figure 6.4.

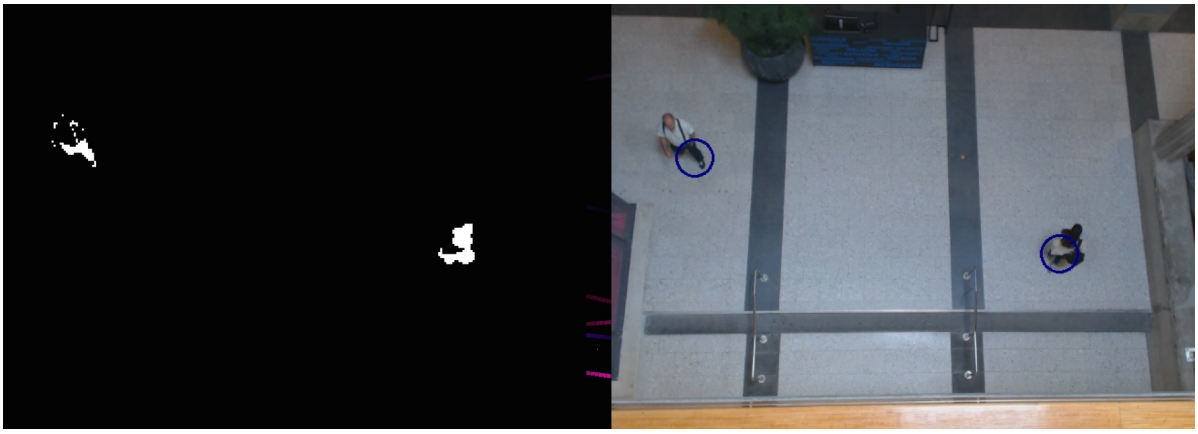


Figure 6.4: Example of the result of background subtraction on the left with the original frame including blue circles showing the final detection on the right.

Blob Detection

One of the encountered problems with this method is that people are not always detected as one blob. For example, a person wearing a t-shirt that has a very similar color to the background will be represented as multiple separate blobs for the person's head, arms and legs leaving a black gap where the shirt was supposed to be. Another problem is that small disturbances in a frame can be detected as foreground and are represented as a blob as well. These issues can be solved by tweaking the parameters of the blob detection.
The two most important parameters are minimum blob area and minimum distance between blobs. Defining a minimum area prevents smaller blobs that cannot be a person from being detected. However, this area cannot be smaller than the area of hands, heads or feet because these parts still need to be detected in case of similar colors of background and clothes. An example of this can be seen in figure 6.5. Defining a minimum distance between blobs makes the blob detection group together blobs that are closer to each other than the minimum. This means that the hands, head and legs of the person wearing the similar colored t-shirt are grouped together and he will be detected as a single person.
Because our system needs to be flexible and should work with any camera placed on any height, these parameters are set based on frame resolution and the number of pixels that define one meter on the scene. From these blobs we extract the coordinates of the center. These coordinates are then converted to locations.

Figure 6.5: Three people whose shirt does not get detected. Their head and legs are all separate blobs.

Perspective

We now have the locations of the centers of each blob or set of blobs. However, what we need for our projection is the location of people's feet. If the camera is placed directly above and in the middle of the scene, the part of the blob that is closest to the middle represent people's feet. Therefore, it suffices to apply a small correction on each location to pull it more towards the middle. If the camera is placed more towards one side, the perspective point in the middle moves towards the other side. The same correction can be made towards the shifted perspective point. For this correction, the frame gets divided into nine equal sections. Based on what section the location is part of, a number is added to or subtracted from the x and/or y coordinate. This number is calculated based on the number of pixels per meter and the height of the camera.
After all these steps we have a collection of all locations of people detected in the current frame. Finally, all these locations are stored and passed to the identifier.

6.3.3. Identifier

As mentioned in chapter 5 about system design, each person who walks onto the scene will get one specific (range of) color assigned to them. Therefore, it is necessary to keep track of what person moved where so they can keep the same color. This is the identifiers task. Every frame, the identifier receives new locations of where people have moved. Because it stores

the people who were present in the previous frame, the identifier can match those locations to people.

Matching People

The first part of matching is done by going over all people from the previous frame first and checking which location is closest to each of them. When a close location is found, the person's location gets updated to the new one. When there is no location close enough where the person could have moved to, there are two options. The first one is that the person is standing still but was not detected because they faded into the background. As mentioned before, the background subtraction algorithm adapts to changes. Therefore, if someone is standing on the same location for too long, the algorithm will classify them as background. Because the system encourages movement and people lose their light trails if they stand in one place for too long and because change in brightness or other factors can cause wrongly detected locations in some frames, it is not an option not to remove anybody until they actually leave the scene. Instead, we opted for a countdown timer. When the timer reaches zero, the person has been standing still for too long or is not a person at all which means that this person/object has to be removed.
The other option when no near location is found is that this person left the scene. This is only possible if their previous location was near the border of the frame. If a person was standing near a border within a certain boundary, they will be removed immediately. This boundary is calculated based on the number of pixels per meter and the frame resolution.

Handle Remaining Locations

Of course, there is also the possibility that new people walked onto the scene. Locations that were matched with a person from the previous frame are removed. This means that the remaining locations are new people who joined or disturbances in the scene due to a change of lighting, for example. It is easy to distinguish people from these disturbances as people walk in from the side of frame. This means that most new locations close to a border are people and others that are located more towards the center are presumably not. For all of these locations that are close to the border, a new person is created.

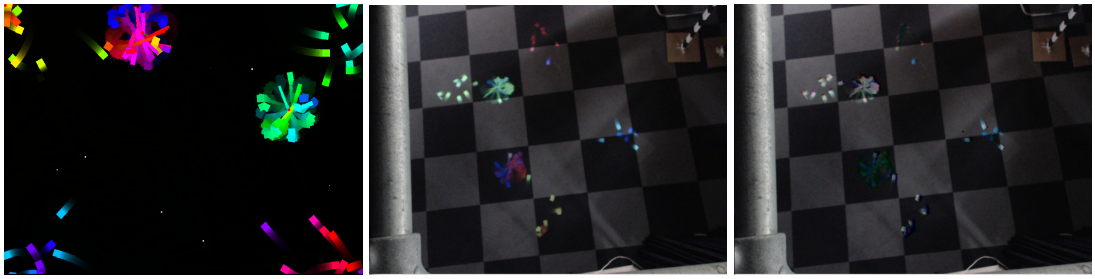
Participants and Bystanders

Also mentioned in system design chapter 5 is the distinction between participants and bystanders. Participants are the people who are located inside of the boundaries of the projection and possibly actively interacting with the system while bystanders are located outside of the projection. The Identifier not only receives the boundary of the frame that can be considered as 'close to the edge' but also the boundaries of the projection. With this information, the identifier can check each location to see if it is inside or outside of the projection and change the type of each person accordingly.

6.3.4. Projection elimination

When trying to identify people on a camera frame while the scene is projected on the floor, the camera captures both the people and the scene projection. When elements on the scene have a size close to the size of a person (seen from birds eye view) or when scene elements cluster towards the size of a person, the elements will be identified as a person due to the chosen method of background subtraction based on movement as described in section 6.3.2.

The chosen method to reduce this detection of scene elements of the projection is projection subtraction discussed in section 3.4.2. This method selects the scene frame of a calibrated time ago due to the delay between projection and video capturing. This selected scene frame is then transformed to a camera sized frame on the calibrated area where the projection is located on the camera frame. This is done using the inverse of the perspective transformation described in subsection 6.2.1. This transformed frame then is subtracted from the camera frame. The result is that the subtracted camera frame has reduced visibility of the scene elements so that detection and identification will be more accurate. An example can be seen in figure 6.6.



(a) Scene frame (b) Unmodified camera frame (c) Projection subtracted frame

Figure 6.6: The projection elimination using projection subtraction reduced the visibility of scene elements so that detection and identification will be more accurate. Note: the projector is rotated about 180 degrees compared to the camera.

The current background subtraction is limited to a calibrated constant factor subtraction of the scene projection. For environments with a diverse floor color spectrum and intensity the subtraction scale should be based on the floor color and intensity.

6.3.5. Extractor

Because the detector and identifier are two separate entities, there needs to be some kind of communication between them. Furthermore, they need some information from the calibration like pixels per meter. We also established that image processing was very slow at first, which was a big problem. The solution for this was down scaling the frame and converting it to gray scale. The detector takes and processes any frame of any size so there needed to be something in between to take care of this. All this is what the extractor does. It receives information about the camera resolution, projection boundaries and how many pixels represent one meter. Based on that information it calculates an appropriate size to resize the frame to and prepares all parameters to pass to the detector and identifier. When the extractor receives a frame, it resizes it and converts it to gray scale. Next, the prepared frame is passed to the detector who returns the detected locations. After this, the extractor passes the locations to the identifier and receives back all the detected people in the frame. The last step is to scale these coordinates back to the original frame size. We now have the right location and information of each person in the frame which can be passed to and used by other entities. Figure 6.7 and 6.8 are examples of the result of people detection.

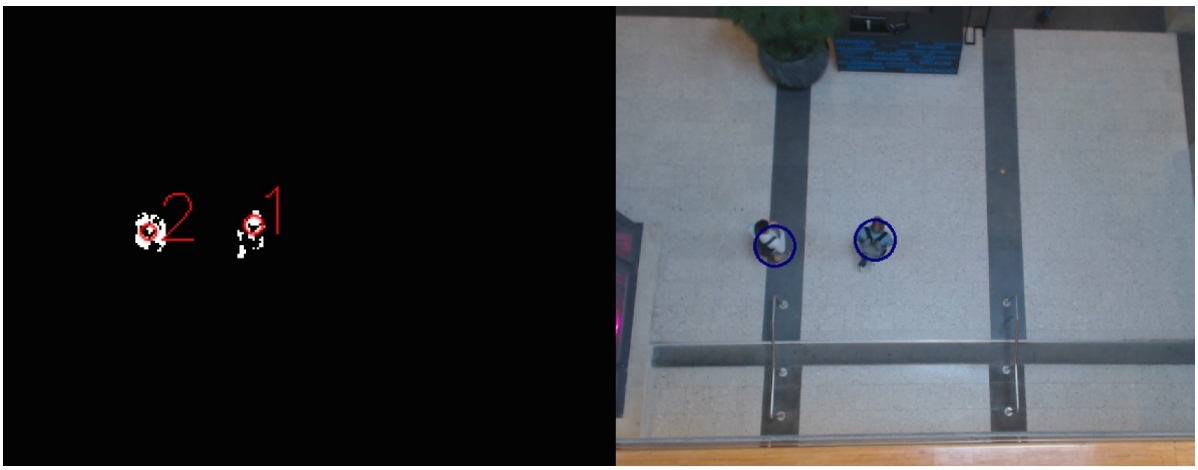


Figure 6.7: First example of the result of image processing.

Figure 6.8: Second example of the result of image processing.

6.4. Scene

The scene component of the software is responsible for turning the input that it gets from the people detection component into the scene. Basically, it needs to implement everything that is specified in the Light Trails section of the System Design.

Since we want it to be possible to easily add more types of scenes to the program without having to change anything existing, we set up an abstract *Scene* class to begin with, that is extended by the specific *LightTrailScene*. The *Scene* class is responsible for everything that happens in every scene, whereas the *LightTrailScene* class is responsible only for Light Trail Scene-specific things, such as drawing of the graphics.

A general scene loop takes the following steps:

1. Check if there is new person data received from the People Detector. If yes, update the people.
2. Check all conditions and execute all actions.
3. Draw all effects.

6.4.1. Event-based design

When taking a quick glance at the way the System Design is structured, it becomes evident that the design is event-based, be it time-based events or input-based events. This boils down to a simple statement: When certain conditions are satisfied, certain actions have to be executed. Every part of the design can be described in such a way, and therefore it seems sensible to create abstract classes that describe the two parts of our statement:

Condition Checks whether a condition is satisfied and creates actions as a consequence.

Action Executes an action, and says when it is done, potentially providing a followup action.

Since this structure is the same for every scene, the *Scene* class is responsible for checking the conditions and executing the actions. The *LightTrailScene* class is then responsible for creating every condition and some initial actions.

Actions do not need to know about each other, and this is enforced in the design. If actions would be able to know about each other, this would surely cause tight coupling between them, with bad maintainability as a consequence.

6.4.2. Entities in the Light Trail Scene

Again following rather simply from our System Design are the entities that play a role in the Light Trail Scene. From the definitions of the Light Trail scene the following classes have emerged, with their respective responsibilities:

LightPerson Keep location, hue and status.

LightTrail Keep location, speed and hue, react to force

LightSource Generate light trails based on a set angle, speed, and hue range.

GravityPoint Keep location and gravity, calculate the force on a light trail.

ColorHole Do all that a GravityPoint does, suck up light trails.

In a Light Trail Scene, many instances of all of these entities exist at once. To keep track of them, we have set up an abstract *Repository* class that provides adding, getting and setting functionality. Because this is and abstract class, we can encapsulate the underlying data structure by using polymorphism. As such, the data structure can be easily replaced when needed.

These repositories keep track of a set of smart pointers towards the entities, because certain actions may need to influence any part of the scene independently. Also, using pointers allows for polymorphism within the entities; color holes may be stored as gravity points, for example.

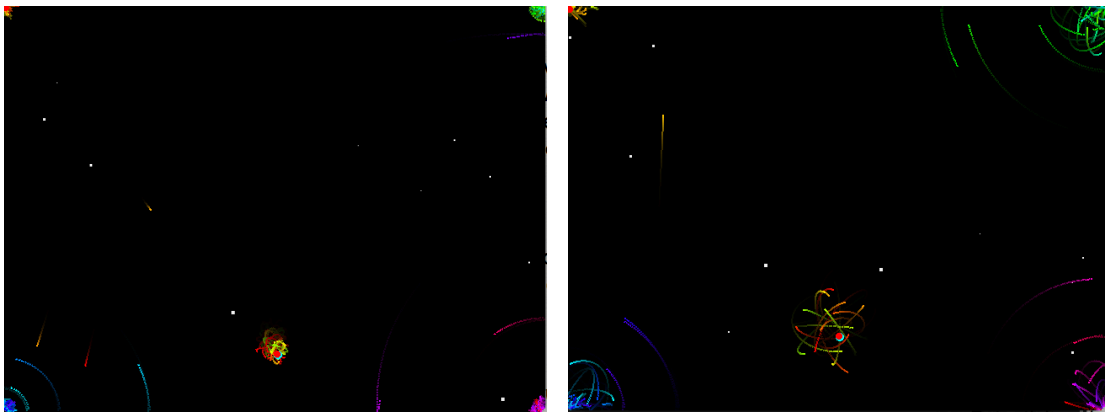
Conditions and Actions clearly show which of these repositories they need access to, simply by specifying them in their constructor. This makes it clear which parts of the scene are accessed where. For example, the *ParticipantGravityPointAction*, which tracks a person and follows it with a gravity points, needs access only to one person, and the repository of gravity points. This is clearly visible in its constructor.

6.4.3. Physics

We want the participants to attract light trails and cause these trails to orbit around them. The first step taken to make this happen was to give each participant a gravity point that exercises gravity on the light trails. We use Newton's gravitational equation for this purpose.

$$F = G\frac{m_1 m_2}{r^2} \tag{6.1}$$

We assume that the mass of a light trail is 1 kg and the mass of the gravity point is defined in the configuration. This works fine in terms of the participants *attracting* the trails, but it will not cause the trails to *orbit* around them; the trails will simply be pulled towards the location of the player, and then be stuck there.



(a) No dead zone. (b) Trails orbiting with dead zone.

Figure 6.9: In both images there is one gravity point that attracts red/yellow trails.

Therefore, some kind of *dead zone* needs to be applied. Within a certain radius from the gravity point, the trails should not be attracted anymore, to keep them from getting stuck in the center. However, applying a complete dead zone does not cause orbit either, and proves to have a very unrealistic effect. Therefore, we haven taken the middle road in our

implementation: the gravity exercised on the light trails *does not get any bigger* within a certain radius. That way, the speed still increases, giving a quasi-realistic effect, and trails will start orbiting.

6.4.4. Limiting the amount of trails

To keep the scene from becoming too busy, we limit the amount of trails that are in the scene. The first approach we have taken to this end is giving the trails a limited lifetime. That way, trails die out while new ones are added, and the amount of trails reaches a stable point.

However, we do not want to limit the amount of trails so much that some people will be walking around without any. Because of that, we increase the amount of trails in the scene with a certain amount whenever a person enters the scene. Then, when they leave the scene, the same amount of trails are removed to keep the scene clean.

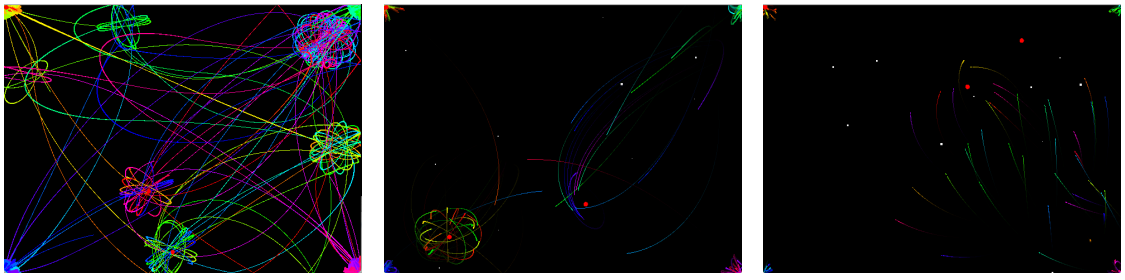
Because we want everybody to be followed by trails, we send every new person n 'initiative trails'. These trails ignore everything except the person, and will be added to the global trails to be affected by everything once they are close enough to the person.

6.4.5. Drawing

After the behind-the-scenes processing of the scene is done, it needs to be drawn. Every action has a list of *effects*. This list may be empty, if the action does not have any visual aspects. When drawing the scene, all effects of all active actions are drawn. Effects, again, are made sure to only know about what they need to know about, as specified in their constructor parameters.

All drawing is done using the SFML library, as stated in chapter 3.

The most important effect in this scene is the Light Trail Effect, which draws the light trails. To create the light trail effect with minimal overhead, we slightly darken (or lighten, if inversion is enabled) everything that was already drawn, and then draw a small square at the head of the light trail. This way, every trail will be lead by a bright head, and have a fading line behind it. Also, trails will have a lasting effect this way, since their trails never completely fade, but their effect is not too intrusive. Of course, we can play with the amount of 'fade' and get different kinds of effects, as seen in figure 6.10.



(a) No fade. (b) Low fade. (c) High fade.

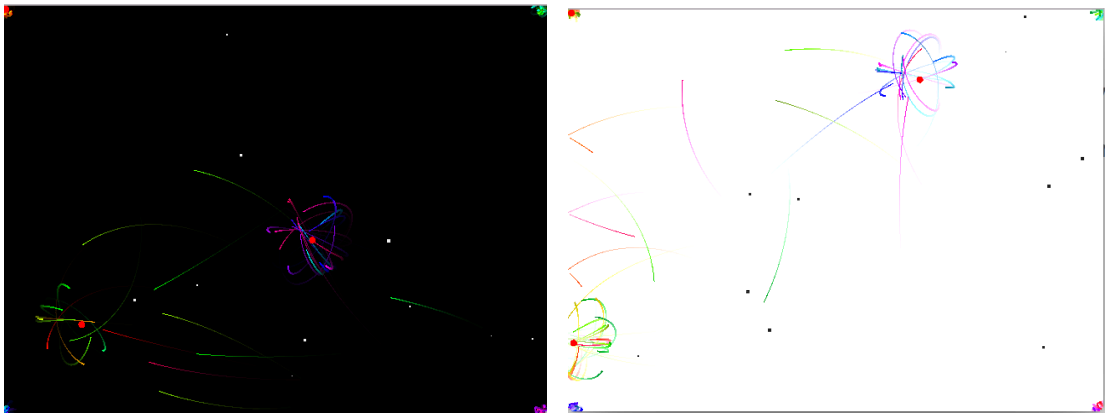
Figure 6.10: The scene with different fade values.

This darkening and lightening is done by using SFML blend modes, which map directly to OpenGL blend modes. To let part of a light trail nearly disappear completely, but not too quickly, we decrease or increase the RGB values with the fade value, dependent on whether or not inversion in enabled, and we decrease the alpha value with the fade value in both cases.

6.4.6. Inversion

At a very illuminated location, such as the TU Delft Aula entrance, described in "Testing" subsection 7.1.2, the normal scene is not very visible. Therefore, we have created the option to invert the scene, to provide more contrast between the trails and the background. In inverted mode, the background is white, the trails fade to white, and colors that are perceived

as lighter are darkened. In figure 6.11 you can see the difference between normal and inverted mode.



(a) Not inverted.

(b) Inverted.

Figure 6.11: The scene in normal mode and inverted mode.

6.4.7. Configuration

In order to avoid magic numbers in the software, and allow for customization of any numbers to improve design reiteration, we have created a configuration file for the scene, that expresses distance values in meters, and time values in seconds. Using the pixels-per-meter value that is calculated by the calibration component, these meter values will then be mapped to pixels. We use YAML 1.0 to describe this configuration.

Using this configuration, we can, for example, set the thickness of the light trails, change how fast the scene fades, as seen in the previous subsection, or change the speed of mixing. By changing these parameters we can adjust the entire experience, and more importantly, optimize it.

Keeping the configuration maintainable has been quite the hassle. At first, due to lack of time, we just used a big data class as the configuration that was passed around to all the places where it was needed. We explicitly did not use the singleton pattern from the start, though, since that would severely reduce testability; a test needs to be able to pass its own configuration object. This data class structure, with an enormous constructor as a result, was not ideal. Therefore, we refactored cohesive parts of the configuration into smaller classes, creating a hierarchical configuration rather than a flat one. Though this increased the number of calls necessary to get to a configuration item, it significantly improved readability and the clarity of the configuration interface.

6.5. Hardware

The minimum requirements for the hardware are based on the previously defined design in section 6.1. Two processes with each a main thread and a thread for communicating the scene frame, 4 CPU cores are required for the machine to run the application at the required speed. Tested on cores with 2.3 Ghz. A slower CPU core is not advised. For graphical processing this is tested on a GeForce GT 720 and Quadro K1000M chip. A slower chip might be possible, but is untested.

For the setup in the TU Delft Aula entrance, described in ”Testing” subsection 7.1.2, the setup includes a projector with 5000 ANSI lumen of an projecting area of 7x4 meter. As referenced in the research in section 3.3.1 this should be enough in an environment with normal daylight and lamp light. However as can be seen in figure 6.12 the scene is not visible, even though it is projected. This is partially because the sensor of the camera is not optimal, because with the human eye it is (barely) visible, but it shows that the defined amount of lumen is not enough. The conditions in the figure are clouded, high summer and around the middle of the day.

This means that it is required to have a better control of the environment. This is currently planned to be done by covering the large windows from where the sun enters the Aula and a white sheet is planned be layed out on the floor to increase reflectivity of the projected scene.

Figure 6.12: The scene is not visible with a 5000 ANSI lumen projector in the clouded summer at midday.

The 1080p Logitech HD Pro c920 Webcam is more than enough, because the people detection part runs on only 384x216 pixels, and in black and white. Any variant camera with an equivalent lens and sensor, with a resolution starting at 400x250 will probably be good enough.

7

Testing

A system like I. M.O.V.E. requires a lot of testing. Even though not everything needs to be exact, tracking of people brings many different possibilities and corner cases and our scene has a wide variation of actions. This means that the system needs to be tested as much as possible to verify all these possibilities. The kind of testing we used most was integration testing. These tests were performed on different systems, with and without camera and beamer and using the different installations mentioned in the first section of this chapter. Apart from or combined with the integration tests, every part had its own testing methods, which will be described in the second part of this chapter along with the complete system tests. The final part is a more detailed description of the testing processes. It explains the results and solutions of the main (scene) integration tests.

7.1. Installations

To be able to test the complete system with a camera faced down from the ceiling and an actual projector, we were granted access to two locations where we were allowed to setp up a full installation of our system.

7.1.1. INSYGHT Lab

From week five on, we have had a full testing setup in the INSYGHT lab on the second floor of the EEMCS building, including a projector and a camera. This setup was not ideal; the scene was about 2 by 1 meters, and the camera hung at a height of about 3.5 meters. Because of the low camera, a maximum of two people could be tracked at once and due to the low projector and small projection field, we were not able to see what was being projected and wether the tracking worked because people's shadows blocked most of it. However, it managed to point out a lot of errors and problems with the system as a whole in relatively early stages of the development, and it was therefore very useful.

7.1.2. Aula

Starting in week 9, we had a full setup in the TU Delft Aula. Both a camera and projector were available at this location and could be placed several meters above the ground. This location was our final location where the system would be displayed for several weeks. Being our primary location, it was, of course, the perfect place to test and tweak the software. Here, it was possible to fully test every part of the system, including communication between parts and system endurance.

7.2. Methods

Each part of the three main parts of I. M.O.V.E. was tested using different methods. Even though these methods were not all completely seperate tests, in this section a description of each method is given for every part seperately.

7.2.1. Calibration

Testing the mapping between camera and projector is primarily done using the installed camera or a webcam. The expected coordinates on the projection are compared to the actual coordinates. Using the projection coordinates set by the user and using the created perspective transformation matrix, the camera frame is transformed to only display a rectangle projection (here after called ”projection image”) which should correspond to what the projector shows. For every coordinate on the camera frame the position is augmented on the camera frame and on the projection frame. The comparison test of expected and actual coordinates is done visually by the developer.

7.2.2. People detection

Videos

The first tests for people detection were done using videos filmed on balconies at the Aula and the industrial design building. These gave an impression of how many people were detected using each feature and different values for each parameter. These videos were very useful during the beginning of the development process to test background subtraction algorithms, basic blob detection and perspective adjustments. Further into the process they were useful to quickly verify the workings of new features before they were tested on a full setup. The only problem was that these videos were not filmed with a stationary camera. This means that movement of the camera could also be detected as general movement and this caused a lot of disturbances in the frame.

Stationary Low Camera

At the first installation location there was a camera available which was attached to the ceiling and facing down. With this stationary camera it was possible to test image processing using a live feed. This solved the problem of movement in the background but brought along some other challenges. The room did not have a very high ceiling wich was a requirement for our system. It was a challenge to tweak the parameters in a way that the system could detect people correctly with both the high camera in the videos or eventually the final location, and the low camera in the lab we worked in. This test setup was sufficient enough to mimic real life situations and challenges we would face in our final large scale setup.

Stationary High Camera

The debug mode of the system, which we used everytime the whole system was tested, displays a window on screen showing the image processing part. This way, we were always able to track how image processing was doing. Therefore, it was easy to identify problems concerning people detection when we tested the system at our final location for the first time. Image processing did fine but it was clear that some parameters needed to be modified and some situations that could not be replicated during the test setup needed to be handled.

7.2.3. Scene

For the scene we use two different kinds of tests: scene integration tests and controller tests.

Scene Integration Tests

The scene integration tests are there to verify that the scene works as a whole: they simulate an input of people, and show the result of that. They cannot be automated, it is up to the

person running the tests to decide whether or not the results are satisfactory. We use these tests mainly to see if overall processes work in their entirety, even under large stress (i.e. many people), and to see if their effects are actually visually appealing. Especially the latter is very subjective and therefore not automatically testable.

Controller Tests

Though integration tests are a good way to see if everything is functional, it does not show whether or not it is interesting for a person participating in it. Because of that, we set up an application wherein people could control a virtual 'person' walking through the scene using an Xbox 360 controller. Using this approach, we were able to get a feel for what walking through such a big scene may be like, and what kind of actions one might take. With this information, we could then reiterate our design.

7.2.4. The complete system

Integration tests

The kind of testing we used most was integration testing. The three main parts were always combined as fast as possible to test the whole system. During these tests there were multiple debugging windows on screen for every part so we would be able to identify quickly where the problem was located as soon as something went wrong. They were very useful to test communication between parts as well. These tests were performed on different systems and locations.

Endurance test

At the end of the project we ran a few endurance tests for the whole system. The system and the machine it runs on have to be able to cope with power loss, because the machine is supposed to run at least for a few months and possiblty even longer. The machine should automatically power on and start the I. M.O.V.E. system after loading. The system also has to be able to cope with unexpected system crashes and auto restart when something happens. Both scenarios have been forcefully tested and the system has had long test runs on different installations to test for stability of the system. All test failures were detected in the span between the five and ten minutes, which is just outside of the normal development test time span.

Unit Tests

We have decided not to use unit testing for this project. Unit testing is mainly important when operations need to have very accurate results. In our case, however, a light trail moving slightly different than it should or a person being detected a few pixels off is not really a problem. Therefore, unit testing would require us to set harsh boundaries on the functionality of the system, creating unnecessary overhead. Also, testing the specific implementation of different actions and conditions is not really important; they are compact, and their code is easily readable. On top of that, they are supposed to be adaptable, and unit testing would lock their interface in place. Most errors occur when the entire system is working as a whole; edge cases generally occur as a result of inconsistencies across multiple actions or conditions. Integration tests suffice to simulate tricky scenarios and reach these edge cases.

7.3. Results

We have run quite a few tests over the course of the project. Now that all methods have been explained, we show an overview of these tests combined and what the results were.

7.3.1. INSYGHT Lab: People

While the INSYGHT lab location had been perfect for testing the calibration and a good tool for testing the image processing, testing the entire system was not really possible in this setting. However, because this was our first full test including camera and projector, some

problems were easy to identify. One of the main problems was the stuttering of the scene due to the low amount of scene frames per second, which was due to the people detector using a high load on the same CPU core as the scene. When this was using multiple processes, we quickly concluded that, if we were to test the system here, we needed something smaller, but still resembling a human. That is when it was suggested that we test with robots.

7.3.2. INSYGHT Lab: Robots

Using robots, we could not necessarily test interaction, let alone user experience, but it definitely made it easier to test the image processing in a live and relative bigger environment. Also, we could use our own judgment to see from the sidelines what things should be changed about the scene.

In the first robot test, we noticed the following two things:

1. A lot of trails were flying around, making the scene too busy. We have fixed this by making all trails that are not attracted by people return to their sources.

2. The scene detected itself a lot. Our first approach to fixing this was implementing projection subtraction, which subtracts the scene, which it knows about, from the input frame before image processing.

In the second robot test, we found the following:

1. The scene was less busy now. Taking trails back to their sources had been successful.

2. Still, the scene detected itself a lot. We tried another approach on top of the projection subtraction approach, which was to only let people be detected if they come in from the side.

The results of these tests helped us to discover and solve some issues but it was still not enough to test participant interaction. Because our client wanted to be able to experience what the interaction was like even though there was no location and stronger projection available we decided to set up a test with controllers.

7.3.3. Controller tests

We have done a couple of tests using controllers, to get a feel for what interacting with the scene is like. This way, we were able to control where each person was going and try out interactions on a large field, just like the final setup. During these tests, we quickly realized that mixing and color holes were not really intuitive enough, and people could easily take away trails from others, leaving some with none.

7.3.4. Aula tests

After fixing all errors and modifying parameters to make our system work properly at the final location, the Aula, we showed it to our client. He gave us feedback on what was needed to improve or change:

1. The scene is not visible enough. This was not something we could change as the location of our setup is very bright. The solution was to cover up the windows directly above so no more light could disturb the scene.

2. People who are standing still need to be detected. As mentioned in section 6.3, a person who is standing still for too long will fade into the background. If the person has not moved by the time the counter has run out, they will not be tracked anymore. During our test session at the Aula it was clear that people were standing still way longer than expected. Therefore, we increased the counter exponentially but did not add the feature never to delete a person standing still to eliminate the risk of small disturbances staying on the scene forever.

3. Tracking is not located on people's feet. Right before the demonstration to our client, we tilted the camera a little more upwards. Now the camera was not completely facing downward anymore which caused change in perspective. We changed the perspective adjustment in the detector accordingly.

Finally, when these issues were solved, we continued testing at the Aula to fully optimize the system for this location.

The environment of the Aula is currently still to be covered from intense sunlight. This currently prohibits the scene from being in the attention of people during most daylight times. The interviews with people to evaluate the system goal (subsection 5.1.2) will be postponed until the environment is finished.

8

Evaluation

Finally, we evaluate our process, design choices, implementation and more through discussion and reflection. We will conclude our project and give a few recommendations.

8.1. Discussion

In chapter 2 we have described the requirements for this project. In this section, we describe how we assess whether or not each requirement was fulfilled, or why we were unable to make such an assessment.

8.1.1. Requirements for the interaction

The goal of the project as described in chapter 1, ”Introduction”, is to let a passerby-crowd generate, interact with and control projected art. As stated in the research section 3.1, ”Interaction design”, the hardest challenge is to change people’s attention as fast as possible to persuade them to participate. This is of high importance, because it is the most significant factor that increases the rate of people participating in the scene. The created system design specifies those different stages and is modeled to add incremental value.

Testing the degree of value for participants is hardly quantifiable, because it is such a subjective matter. The best option to verify this would be surveying the people passing through the area. However, at the time of publishing this report, the single installation, at the Aula, is not yet at its desired potential, because the windows have not been covered yet. Doing any kind of user interviews would not give results representative for the quality of the system. The judgment of whether or not this project has been a success is then simply left with the client.

In a project like this that focuses heavily on interaction, the design needs to be adaptable and, as a consequence, the software needs to be very adaptable. Due to the event-based design, conditions and actions can be added and removed with great ease. Thus, should the project need to be adapted after its conclusion in order to improve the quality of the interaction, that would be relatively easy to do. Therefore, though it is not expected that the project will be unsatisfactory, the software is prepared for the scenario.

8.1.2. Objective goals

While assessing the success of the requirements related to interaction is very difficult, the fulfillment of objective requirements is easier to assess. In this subsection, we will cover these objective requirements.

Tracking changing locations of crowds/individuals

The tracking of people performs very well in the Aula setting. Using testing videos we can compare the results with our human eye, and see that the tracking is very satisfactory. Therefore we call this requirement fulfilled.

Displaying projective art controlled by the crowd
We specified the way this goal was going to be executed in the System Design in chapter 5. By executing and testing the software, we can see if all the events described in the system design actually happen. It is clear to see when doing such tests that the system design is in fact fulfilled by the software. Because of that, we will also say that this requirement has been fulfilled.

No constraints for participation
As a natural result from good people tracking and displaying of the art, people do not need anything other than to be in view of the camera to participate. Neither is the participant forced at any time in the scene to fulfill an action before being able to control the scene. Therefore, this requirement is fulfilled.

8.2. Reflection

In this section we look at our own experience with the project, what we learned and where we applied the knowledge we have gained from the computer science bachelors.

8.2.1. Project structure
Due to the freedom of the project, we had to enforce structure upon ourselves. For the first time in the study program, we were responsible for the entire project management ourselves. This has been a very valuable experience. It has taught us responsibility, and perhaps more importantly, pro activity. If we wanted something to happen we had to make sure that it happened. Thankfully, we have had a lot of help from our client, Rafael Bidarra, when it came to acquiring a location and the gear required for a full setup. Without his excitement and connections it would have been very unlikely this would have happened within the frame of the project. This teaches us that connections are important, and that we should try to create and maintain them, should we ever want to achieve something similar.

8.2.2. Interaction design
The requirement for human interaction and encouragement thereof is deeply ingrained in this project. Interaction design has played a role in other projects in the bachelors, but it has not been as important as it was in this project. This has proven to be quite a challenge for us. As computer science students, our expertise is creating software that does what is has to do. Part of this expertise is to translate clients' requirements and loose defined scope into software. This is where the system design comes in. In this project the creation of the system design was relatively difficult, since we had to base it on a part of reality that is not at all clearly defined: human behavior. Thankfully, we are far from the first people to come across such issues, and literature was available on the subject. However, translating this literature into a design for our specific case was quite the challenge still.

8.2.3. Application of knowledge
The part of our curriculum that is most prominent in nearly any software project is the Software Engineering Methods course. This course has taught us how to create good software that adheres to the SOLID principles. We have applied our knowledge gained about this throughout the entire project, and have been above average successful at that, judging from our first SIG rating. In the development of the people detector, knowledge from the Image Processing and Multimedia Analysis elective courses, which two of us have followed, has been very applicable. Our prior experience with OpenCV and image processing techniques made it significantly easier for us to get a good basis going for this part of the software.

8.3. Conclusion

In the past ten weeks, we have built an interactive system that uses nothing more than a camera, a projector and a general computer. People do not need anything other than to be

physically present in order to participate.

We have done research on the methods we could use to fulfill the project requirements, we have planned out the structure of the project, and we have extensively tested the system. Using projection subtraction, background subtraction and blob detection, we have implemented a real-time top-down people detector, along with an identifier that keeps track of which person is whom. We have created a system design that aims to encourage interaction, and by modeling the software design of the scene to match the event-based design of the system, we have made the scene adaptable and maintainable. Though it is very difficult to assess whether or not the interaction design was successful, our system is made in such an adaptable way that it can always be changed to improve interactivity, should this be necessary. Also, the system can be run in any environment with limited light, large open floor and high enough ceiling to show a large projection, because of its calibration component.

Through all these efforts we have successfully brought I. M.O.V.E into the world. We have created a low-budget interactive experience for everybody to enjoy. I. M.O.V.E. literally shows people their normal surroundings in a different light and gives them the ability to control the light, which allows them to be entertained, to explore and to express themselves.

8.4. Recommendations

In this project, we have not only created a setup that allows people to interact with each other in a creative way, we have also laid the foundation for more of these kinds of projects to come. The system is very adaptable and a new scene would be interchangeable with our Light Trail Scene. A setup containing just a camera, a projector and a computer is relatively cheap for such a large interactive experience. Therefore we recommend that this project is expanded upon, so that many more interactive experiences such as this one may be created.

Another recommendation is to actually assess the interaction design when the windows of the Aula are covered by interviewing people who participated in the scene. This assessment will enrich the known value of the system and this feedback can be transformed and embedded into the system for possible enhanced value.

A

Infosheet

Procedural Real-time Crowd Art

Product I. M.O.V.E. - Interactive Moment Of Visual Expression

Client Computer Graphics and Visualization group, Intelligent Systems department, Faculty EEMCS, TU Delft

Presentation date Friday July 1st 2016

Public exposition location TU Delft Aula entrance

A.1. Project

Challenge Designing and building an interactive system based on motion tracking which allows people to generate, control and interact with projective art.

Research Methods for stimulating crowd participation, people detection and tracking, drawing a scene and calibrating the setup.

Process Upfront general planning and general design, separate responsibilities, agile development with time boxes of weeks and days and system testing. Unexpected challenges interprocess communication and projecting in a bright space were solved resp. by shared memory mapping and controlling the environment

Product An interactive light trail scene for passerby-crowd with colorful light trails following your movements and a variation of events to discover.

Outlook Deployed at TU Delft Aula entrance.

A.2. Team

Rafael Bidarra R.Bidarra@tudelft.nl, client contact, CGV group, TU Delft

Nestor Salamon N.ZiliottoSalamon@tudelft.nl, coach, CGV group, TU Delft

Marie Kegeleers Marie@Kegeleers.be, team member
Activities: planning, people detection

Gerbert van Nieuwaal G.vanNieuwaal@McDuck.eu, team member
Activities: communication, calibration

Wouter Posdijk W.Posdijk@gmail.com, team member
Activities: general design, scene

This final report can be found at: `http://repository.tudelft.nl`

B

Original project description

[9] Experimental interactive experiences have sometimes been set up in public places but they mostly limit this interactivity to a few individuals. This project aims at developing interactive image processing techniques for enabling large, moving crowds to contribute to the real-time creation of procedural projection art (as inspiration only, see e.g. the excerpt of Seventh Sense at `https://youtu.be/iQlDEPLHPyQ`).

Context

Large infrastructures like stations and malls often contain huge surfaces (e.g. floors and walls) where passer-by crowds could both generate, control and enjoy displayed projective art. The experimental outcome of this project aims at being actually used in such a setting.

Assignment

In this project we will investigate how to combine the real-time analysis of high-resolution video content form a crowded scene with its efficient transformation into an artistic projective stream to some output scene. The main challenges to approach include the development of real-time methods for, among (many) other things:

- procedurally generating abstract geometry from data on actual people's (changing) locations
- letting trails left by walking people creatively intermix and interact (e.g. based on color, speed, direction,…)
- using heatmap and other crowd analytics information for artistic purposes

In addition, the project will explore some more directly game-oriented crowd interaction, like e.g. leading people to (possibly collectively) perform some action(s) and/or adjust their behavior, either for entertainment or for some gamified purpose(s).

Requirements

Good programming skills, in particular on CG topics, are a pre.

C

Software Improvement Group

The chapter attaches the first feedback from the Software Improvement Group (SIG), covers the evaluation for the project and attaches the second feedback from SIG.

C.1. First feedback

De code van het systeem scoort bijna 4.2 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code bovengemiddeld onderhoudbaar is. De hoogste score is niet behaald door een lagere score voor Unit Interfacing.

Voor Unit Size wordt er gekeken naar het percentage code dat bovengemiddeld lang is. Het opsplitsen van dit soort methodes in kleinere stukken zorgt ervoor dat elk onderdeel makkelijker te begrijpen, te testen en daardoor eenvoudiger te onderhouden wordt. Binnen de langere methodes in dit systeem, zoals bijvoorbeeld de 'Calibration::createFromFile'-methode in Calibration.cpp, zijn aparte stukken functionaliteit te vinden welke ge-refactored kunnen worden naar aparte methodes. Commentaarregels zoals bijvoorbeeld '// read frames_projector_ camera_delay from yml using OpenCV FileNode; default if not existing' en '// retreive camera resolution from OpenCV VideoCapture' zijn een goede indicatie dat er een autonoom stuk functionaliteit te ontdekken is. Het is aan te raden kritisch te kijken naar de langere methodes binnen dit systeem en deze waar mogelijk op te splitsen.

Voor Unit Interfacing wordt er gekeken naar het percentage code in units met een bovengemiddeld aantal parameters. Doorgaans duidt een bovengemiddeld aantal parameters op een gebrek aan abstractie. Daarnaast leidt een groot aantal parameters nogal eens tot verwarring in het aanroepen van de methode en in de meeste gevallen ook tot langere en complexere methoden. 'LightTrailConfiguration'-methode in LightTrailConfiguration.cpp bijvoorbeeld heeft 21 parameters. Het Unit interfacing score can beter worden als de relevante methode parameters door nieuwe classes ingekapseld worden: bijvoorbeeld een 'Gravity' class die '_participantGravity', '_bystanderGravity', and '_bystanderGravityDelay', etc inkapselt.

Om bij toekomstige aanpassingen duidelijker te maken wat er precies meegegeven moet worden aan deze methodes is het aan te raden een specifiek type te introduceren voor deze concepten.

Over het algemeen scoort de code bovengemiddeld, hopelijk lukt het om dit niveau te behouden tijdens de rest van de ontwikkelfase.

Als laatste nog de opmerking dat er weinig (unit)test-code is gevonden in de code-upload. Het is sterk aan te raden om in ieder geval voor de belangrijkste delen van de functionaliteit automatische tests gedefinieerd te hebben om ervoor te zorgen dat eventuele aanpassingen niet voor ongewenst gedrag zorgen.

C.2. Evaluating first feedback

Our code is above average maintainable. This is an achievement with which we are happy. Before we submitted the code to SIG, we knew that the classes 'LightTrailConfiguration' and 'Calibration' were too large. These classes were ment to only hold data and therefore assessed the risk to be low for our project and thereby putting our focus on other parts. The change in design has been postponed until the second submission to SIG.

To reduce the Unit Size the class 'Calibration' (renamed to 'ImoveConfiguration') is subdivided into classes 'CameraConfiguration', 'ProjectorConfiguration' and 'ProjectioneliminationConfiguration'. Corresponding methods such as 'Calibration/ImoveConfiguration ::createFromFile' are hereby split up into their own method '<class>::createFromNode'.

To reduce the Unit Interfacing the class 'LightTrailConfiguration' has been subdivided into classes 'TrailConfiguration', 'GravityConfiguration' and 'EffectConfiguration'. These classes have again been subdivided into in smaller classes. Hereby the amount of parameters of the constructor has been reduced and become manageable.

We have chosen to use no (unit)test-code in our project. The risk is that changes to the code may result in unexpected behavior of the application. More information about this chosen method can be found in subsubsection 7.2.4.

C.3. Second feedback

In de tweede upload zien we dat het codevolume sterk is gegroeid, terwijl de score voor onderhoudbaarheid ongeveer gelijk is gebleven.

In de feedback op de eerste upload werden Unit Size en Unit Interfacing als verbeterpunten genoemd. Bij Unit Interfacing zien we een zichtbare verbetering, zowel in het refactoren van de bestaande code als in de nieuwe code, maar bij Unit Size is dat anders. Jullie hebben daar weliswaar de genoemde voorbeelden aangepast, maar die verbeteringen worden weer grotendeels ongedaan gemaakt doordat de nieuwe code ook weer nieuwe lange methodes bevat. Daarnaast is aan bestaande methodes nieuwe functionaliteit toegevoegd, waardoor die methodes ook langer zijn geworden (CalibrationProjectionWindow::drawImage is hier een voorbeeld van). Dat soort situaties is vrij normaal, maar het is de bedoeling om tijdens het aanpassen dan tegelijk na te denken of die methode eventueel kan worden gerefactored.

Jullie hebben zoals gezegd vrij veel niewue code toegevoegd, maar daar staat tegenover dat er nog steeds weinig testcode is. Dit is wat ons betreft een gemiste kans, aangezien je na elke aanpassing het hele systeem nu met de hand moet gaan testen.

Uit deze observaties kunnen we concluderen dat een deel van de aanbevelingen van de vorige evaluatie zijn meegenomen in het ontwikkeltraject.

D

Configuration

This appendix covers the configurable parameters for the calibration and the lighttrail scene and gives their unit and definition.

D.1. Calibration

This section shows the configurable parameters for the calibration and gives their unit and definition.

Parameter	Unit	Definition
General		
Debug_mode	bool	When 1 shows debug images
Camera		
Camera_device	unsigned char (id)	Operating system defined id for camera
Resolution_camera	tuple in pixels (width, height)	Resolution of the camera
Meter_camera	float (pixels/meter)	Amount of pixels on a camera frame per measured meter
Projector		
Fullscreen_projector	bool	When 1 show projector fullscreen on most left screen, when 0 as window
Resolution_projector	tuple in pixels (width, height)	Resolution of the scene
Maximum_FPS_scene	unsigned int (frames / second)	Limits the frames per second of the frame
Meter_projector	float (pixels / meter)	Amount of pixels on a scene frame per measured meter
Projection		
Projection_top_left	tuple in pixels (x, y)	Coordinate on camera frame of top left corner of the projection
Projection_top_right	tuple in pixels (x, y)	Coordinate on camera frame of top right corner of the projection
Projection_bottom_left	tuple in pixels (x, y)	Coordinate on camera frame of bottom left corner of the projection
Projection_bottom_right	tuple in pixels (x, y)	Coordinate on camera frame of bottom right corner of the projection
Projection subtraction		
Projector_background_light	float	Intensity to subtract projection
FPS_capture_scene	unsigned int	Amount of scene frames to catch per

Parameter	Unit	Definition
Projection subtraction (continued)		
Factor_resize_capture_scene	unsigned int (frames / second)	Higher means faster receiving by people detection, but lower accuracy second for projection subtraction
Frames_projector_camera_delay	unsigned int (scene frames)	Amount scene frames delay before subtracting projection
Iterations_delay_peopleextracting	unsigned int	Aligning people detector by amount delay iterations

D.2. Lighttrail

This section shows the configurable parameters for the lighttrail scene and gives their unit and definition.

Parameter	Unit	Definition
General		
Resolution	tuple(pixels)	The resolution of the screen.
Meter	pixels	The amount of pixels that represents a meter in real life.
Light Source Hues		
TopLeftHue	tuple(degrees)	The hue range of the top left corner.
TopRightHue	tuple(degrees)	The hue range of the top right corner.
BottomRightHue	tuple(degrees)	The hue range of the bottom right corner.
BottomLeftHue	tuple(degrees)	The hue range of the bottom left corner.
Light Sources		
SendOutDelay	seconds	The amount of time the light sources wait before sending out the next trail.
SendOutSpeed	tuple(m/s)	The range of speeds the trails send out trails at.
TrailCap	int	The maximum amount of trails in an empty scene.
SourceHueChangeRange	meters	The radius around a light source wherein a person changes color.
Participant		
ParticipantGravity	float	The gravity of a participant.
ParticipantAntiGravity	float	The antigravity of a participant.
ParticipantGravityRange	meters	The radius wherein a particpant attracts trails.
ParticipantGravityDistance	meters	How far in front of a participant their gravity point is placed.
ParticipantMovedThreshold	m/s	How fast a player needs to move to alter what their 'front' is.
ParticipantSideThreshold	meters	How close a person needs to be to the side before their gravity point is not placed in front anymore.
Bystander		
BystanderGravity	float	The gravity of a bystander.
BystanderGravityDelay	seconds	The time it takes to switch the bystander phase (see system design).
BystanderGravityRange	meters	The radius wherein a participant attracts trails.

Parameter	Unit	Definition
Alternating Gravity Points		
AlternatingGravity	float	The gravity of an alternating location gravity point.
AlternatingGravityDelay	seconds	How long the alternating gravity point waits before changing its location.
AlternatingGravityRange	meters	The radius wherein an alternating gravity point attracts trails.
Dead Zone		
ProximityRange	meter	The radius of the gravity dead zone.
ProximityModifier	float	The number the gravity is multiplied by within the dead zone.
Light Trails		
SidesEnabled	bool	Whether or not the trails bounce off the sides of the scene.
SpeedCap	m/s	The maximum speed of a light trail.
TrailThickness	m^2	The total area of the head of a trail.
TrailMaxLength	meters	The max length of the head of a trail.
Drawing		
Fade	uint8	How much the trails fade every frame.
Inverted	bool	Whether or not the colors should be inverted.
Mixing		
MixingSpeed	degrees/s	How quickly mixing goes at the largest distance.
MixingDistance	meters	The distance wherein people engage in mixing.
MixingRevertTime	seconds	How long it takes to revert mixing.
MixingTrailRange	meters	The radius wherein trails are affected by mixing.
MixingEffectThickness	meters	How thick the mixing effect is.
Explosion		
ExplosionAntigravity	float	The antigravity of an explosion effect.
ExplosionGravity	float	the gravity of an explosion effect.
ExplosionExTime	seconds	The duration of the antigravity of an explosion.
ExplosionInTime	seconds	The duration of the gravity of an explosion.
Light Source Gravity		
LightSourceGravity	float	The gravity of a light source.
LightSourceGravityRange	meters	The radius wherein a light source attracts trails.
Color Hole		
ColorHoleDelay	seconds	How long it takes before a new player is selected as the color hole.
ColorHoleGravity	float	The gravity of a color hole.
ColorHoleGravityRange	float	The radius wherein a color hole attracts trails.

Parameter	Unit	Definition
Color Hole (continued)		
ColorHoleRange	float	The radius wherein a color hole consumes trails.
ColorHoleEffectThickness	meters	How thick one of the circles of the color hole effect is.
ColorHoleEffectPeriod	seconds	The time between the creation of each color hole effect circle.
Stars		
StarAmount	integer	The amount of stars in the background.
StarSpeed	m/s	The maximum speed of the background stars.
Initiation		
StandingStillFadeTime	seconds	How long it takes for a person that is standing still to lose all gravity.
InitiateTrailRange	meters	How close initating trails need to be to a person to be added to the scene.
BystanderInitiateTrails	integer	How many trails are created for a bystander.
ParticipantInitiateTrails	integer	How many trails are created for a participant.
Light Source Effect		
SourceTrailAmount	integer	How many trails are in the light source effect.
SourceTrailPlacementRange	meters	The radius wherein trails can be placed for the light source effect.

E

Poster


I. M.O.V.E
BACHELOR END PROJECT COMPUTER SCIENCE
I. M.O.V.E. stands for Interactive Moment Of Visual Expression and is an interactive system based on motion tracking which aims at entertaining people in public spaces using projections. People tend to stay within their comfort zone and do only that which they aim to do: take a break, wait for something or someone, or get to their destination. This installation aims to pull them out of their comfort zone, and let them show their true, creative colors to the people around them by letting people control projected art and allowing them to interact with each other. We want them to be inspired to be themselves, connect with other people, and turn a moment of inactivity into a social interactive experience.
Step onto the scene and get immersed into a moment of self expression, interaction and fun
I. M.O.V.E. is the Bachelor project of Marie Kegeleers, Gerbert van Nieuwaal and Wouter Posdijk under supervision of Rafael Bidarra and Nestor Salamon - TU Delft faculty of EEMCS

Bibliography

[1] The freeglut project :: About. URL http://freeglut.sourceforge.net/. Accessed: 25-4-2016.

[2] Ccv. http://libccv.org/, . Accessed: 25-4-2016.

[3] Simple directmedia layer, . URL https://www.libsdl.org/. Accessed: 25-4-2016.

[4] Opencv. http://opencv.org/. Accessed: 25-4-2016.

[5] Simple and fast multimedia library. URL http://www.sfml-dev.org/. Accessed: 25-4-2016.

[6] Simplecv. http://simplecv.org/. Accessed: 25-4-2016.

[7] Dynelle Abeyta. What is pair programming? URL http://www.galvanize.com/blog/what-is-pair-programming/#.V2k-kO2lilM. Accessed: 21-6-2016.

[8] AENC. Lichtopbrengst. URL http://www.beamerplanet.nl/lichtopbrengst.html. Accessed: 29-4-2016.

[9] Rafael Bidarra. Procedural real-time crowd art. URL https://bepsys.herokuapp.com/projects/view/165. Accessed: 29-4-2016.

[10] Aaron F Bobick, Stephen S Intille, James W Davis, Freedom Baird, Claudio S Pinhanez, Lee W Campbell, Yuri A Ivanov, Arjan Schütte, and Andrew Wilson. The kidsroom: A perceptually-based interactive and immersive story environment. *Presence: Teleoperators and Virtual Environments*, 8(4):369–393, 1999.

[11] boost c++ libraries. Chapter 15. boost.interprocess - 1.54.0. URL http://www.boost.org/doc/libs/1_54_0/doc/html/interprocess.html. Accessed: 21-6-2016.

[12] H. Brignull and Y. Rogers. Enticing people to interact with large public displays in public spaces. *Human-Computer Interaction INTERACT '03*, pages 17–24, 2003.

[13] ISTQB Exam Certification. What is system testing? URL http://istqbexamcertification.com/what-is-system-testing/. Accessed: 21-6-2016.

[14] Nomis Ch, Shervin Emami, Vincent Ravier, Gus K Lott, Sanjiv K Bhatia, and Peb Aryan. opencv-users - imshow(), waitkey() from within a thread –> not updating the display. URL http://opencv-users.1802565.n2.nabble.com/imshow-and-waitkey-from-within-a-thread-gt-not-updating-the-display-td6918493.html. Accessed: 21-6-2016.

[15] Elizabeth F Churchill, Les Nelson, Laurent Denoue, Jonathan Helfman, and Paul Murphy. Sharing multimedia content with interactive public displays: a case study. In *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques*, pages 7–16. ACM, 2004.

[16] Coolblue. Advies over beamers = beamercenter.nl. URL http://www.beamercenter.nl/advies/beamers.html. Accessed: 29-4-2016.

[17] R. Pausch D. Maynes-Aminzade and S. Seitz. Techniques for interactive audience participation. *ICMI*, 2006.

[18] J. W. Davis and A. F. Bobick. The representation and recognition of human movement using temporal templates. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 928–934, Jun 1997. doi: 10.1109/CVPR.1997.609439.

[19] ENGIE Electrabel. Electrabel tv spot: eindejaarscampagne (nl). URL https://www.youtube.com/watch?v=nCf8E6QrGqc. Accessed: 25-4-2016.

[20] Git. Git. URL https://git-scm.com/. Accessed: 21-6-2016.

[21] GitHub. Github. URL https://github.com/. Accessed: 21-6-2016.

[22] SIG (Software Improvement Group). Getting software right. URL https://www.sig.eu/nl/. Accessed: 21-6-2016.

[23] Steffan Harries. Continuous integration. URL http://books.stuartherbert.com/if-i-knew-then/continuous-integration.html. Accessed: 21-6-2016.

[24] Mark Johnson. What is a pull request? URL http://oss-watch.ac.uk/resources/pullrequest. Accessed: 21-6-2016.

[25] S. Robertshaw K. O'Hara, M. Glancy. Understanding collective play in an urban screen game. *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work*, 2008.

[26] Tim Maly. Using motion capture and code, to turn gymnasts into data art. URL http://www.fastcodesign.com/1669232/using-motion-capture-and-code-to-turn-gymnasts-into-data-art/1. Accessed: 25-4-2016.

[27] MindsTools. Timeboxing - time managment training from mindtools.com. URL https://www.mindtools.com/pages/article/timeboxing.htm. Accessed: 21-6-2016.

[28] Daniel Moreno and Gabriel Taubin. Simple, accurate, and robust projector-camera calibration. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 464–471. IEEE, 2012.

[29] Elements of Design LLC. 'spawn glow free' (art, fireworks and light-show)' in de app store. URL https://itunes.apple.com/nl/app/spawn-glow-free-art-fireworks/id377851369?mt=8. Accessed: 29-4-2016.

[30] Kenton O'Hara, Maxine Glancy, and Simon Robertshaw. Understanding collective play in an urban screen game. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 67–76. ACM, 2008.

[31] OpenCV. Opencv: Camera calibration, . URL http://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html#gsc.tab=0. Accessed: 25-4-2016.

[32] OpenCV. Opencv: Introduction to opencv-python tutorials, . URL http://docs.opencv.org/3.1.0/d0/de3/tutorial_py_intro.html#gsc.tab=0. Accessed: 29-4-2016.

[33] opencv dev team. Geometric image transformations - opencv 2.4.13.0 documentation, . URL http://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html#getperspectivetransform. Accessed: 21-6-2016.

[34] opencv dev team. Operations on arrays - opencv 2.4.13.0 documentation, . URL http://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html#void%20perspectiveTransform(InputArray%20src,%20OutputArray%20dst,%20InputArray%20m). Accessed: 21-6-2016.

[35] M. Piccardi. Background subtraction techniques: a review. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 4, pages 3099–3104 vol.4, Oct 2004. doi: 10.1109/ICSMC.2004.1400815.

[36] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara. Detecting moving shadows: algorithms and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):918–923, July 2003. ISSN 0162-8828. doi: 10.1109/TPAMI.2003.1206520.

[37] Quora. What is object-oriented programming? - quora. URL https://www.quora.com/What-is-object-oriented-programming. Accessed: 21-6-2016.

[38] scottmahoy. Access - an interactive art installation by marie sester. URL https://www.youtube.com/watch?v=678EaXPekFo. Accessed: 25-4-2016.

[39] Marie Sester. Access - an interactive art installation by marie sester, 2003. URL http://accessproject.net/. Accessed: 25-4-2016.

[40] SFML. Opening and managing a sfml window (sfml / learn / 2.3 tutorials). URL http://www.sfml-dev.org/tutorials/2.3/window-window.php. Accessed: 21-6-2016.

[41] Jos Stam. Real-time fluid dynamics for games. *Game Developer Conference*, 2003.

[42] Trello. Trello. URL https://trello.com/. Accessed: 21-6-2016.

[43] xboxde. Kinect interactive art installation - full version. URL https://www.youtube.com/watch?v=wKkMPYmdFHI. Accessed: 25-4-2016.

[44] He Zhao. hezhao/calibrator: Calibrate your projector-camera system on a single click. URL https://github.com/hezhao/Calibrator. Accessed: 25-4-2016.

[45] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773 – 780, 2006. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/j.patrec.2005.11.005. URL http://www.sciencedirect.com/science/article/pii/S0167865505003521.