# *Human Pencils: Drawing your attention*

| **Names:** | **Studentnumbers:** |
|---|---|
| **Liam Clark** | **4303423** |
| **Mitchell Hoppenbrouwer** | **4243889** |
| **Martin Koole** | **4217500** |
| **Jorai Rijsdijk** | **4282264** |

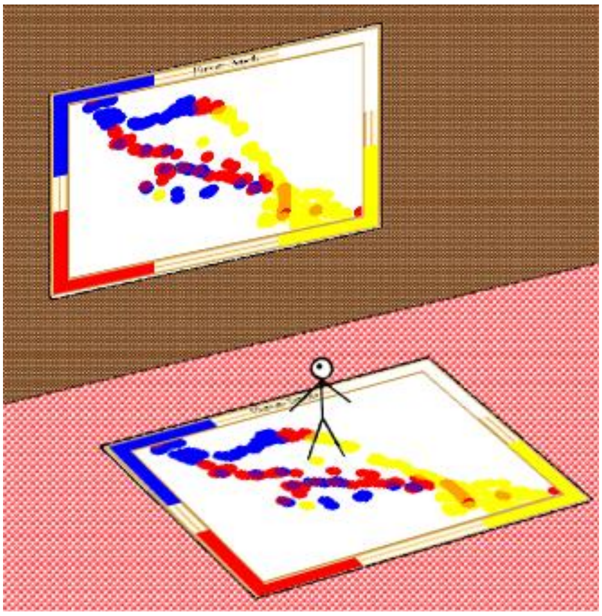
**Course:**
**Context Project Games group 5**

**Date:**
**June 25, 2015**

TU Delft Delft University of Technology

# Table of contents

# 1. Introduction



The goal for this project is to develop a game which is meant to be played in busy, public areas. The game should make use of light projections, let it be on walls or floor. Ultimately, the overall product must be attractive to people who are passing by, so that they'll easily join the game.

Collaboration is something on which the game heavily relies, so rather than focussing on competitive play style, it should be mainly about working together, without the pressure to perform.

The game is particularly meant for people who have some spare time on their hands, people who are searching for a way to kill that time in an exciting way. It must be a unique experience, allowing players to release their creativity and simply have fun.

In order to satisfy these needs, we came up with the idea to create a game that allows people to draw on a virtual canvas, by using their own movement.

Multiple participants are able to 'paint' on the floor by they roam freely through the playing field, leaving a trail behind at will. These coloured strokes will remain in place, as if real paint is applied to the canvas.

A first projection renders these strokes on the ground, giving players the full drawing experience. A second projection will be used to display the same process on a nearby wall. This provides players with a clear view on what they've created so far, enabling them to judge their progress, as well as adjusting the picture more easily.

Movement will thus be a very important aspect of the game, allowing people to fully engage in the physical space and interact with virtual objects.

In the end, we aim for the game to be enjoyable for various types of players, no matter what age or interests they have. It should be something for all to play, and should bring people together by letting them work on their own piece of art, just by playing *Human Pencils*.

This document describes the procedures taken to achieve the final product, focusing on development, implementation, testing and validation. In short, this document first gives an overview of the developed and implemented software product, followed by a reflection on the product and process from a software engineering perspective. Next, there will be a description of the developed functionalities the game provides, and a special section based on interaction design.

Furthermore, we will give an evaluation of the functional modules and the product in its entirety, including the failure analysis.

Finally, this document will provide an outlook.

# 2. Overview of the developed and implemented software product

In order to achieve the desired result as introduced in the previous section, we created a game called *Human Pencils*. This game allows people to play without any form of pressure, and lays emphasis on the aimed main collaboration aspect.

## 2.1 Setup

The playing field is created by the view of a camera fixed to the ceiling, rendering it to look straight down onto the floor. Everything that lays within its view is set to be the part of the canvas, thus the playing field, creating  a rectangular shape.

Furthermore, the game makes use of two beamers. The first is positioned in the same way as the camera; attached to the ceiling and aiming directly downward. This creates the projection on the floor, allowing players to clearly see the paint trail they leave behind upon moving, as well as giving a more detailed view on the painting.

A second beamer is used to create an equal projection on the wall, used to grant players a clear view on their overall progress. It provides a better view on the result so far, also indicating blank spots or slight errors to the participants.

## 2.2 Gameplay

Whenever people want to join the game, all they have to do is step onto the canvas and start roaming the playing field. The assigned colour may vary, depending from which direction the player entered the screen.

Players have the ability to increase or decrease the width of the trail they leave behind, simply by changing the radius of the span created by their arms.

When players want to change their current colour, they can walk to each of the four coloured areas on the canvas' frame and step on top of them. Players can choose between the three primary colours red, yellow and blue, but also an eraser option so they can make adjustments or undo mistakes.

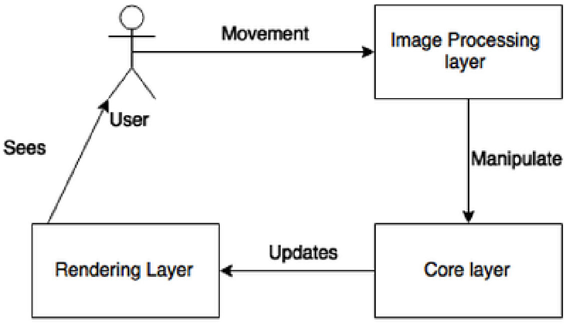
Furthermore, players can combine colours by walking over a previously drawn trail with a different colour. This may result in the secondary colours, namely orange, purple and green. When it so happens that all primary colours are drawn on the same spot, black will appear.

Players can leave the game by merely stepping away from the camera's view, without changing the state of the game itself.

In chapter 4. Developed functionalities, we will give a more detailed description of these features.

## 2.3 Architecture

For the architecture of our game, we use a model-view-controller design pattern. In our design the rendering layer is our view. The image processing layer is the controller and the core layer is the model. We use the model-view-controller design because it fits the purpose of the game well.



What we wanted is a game that uses the player’s position to determine what colour needs to be drawn and then show that to the player. This works well with model-view-controller, because the player position represents the input. The computing of colour, the model. And the displaying of the results, the view.

## 3. Reflection on product and process

This section discusses the development process of the product, and the final result emerging from it. We focus on the particular steps that defined this overall process, indicating what went great, which aspects gave us a hard time and how we managed to fix problem we ran into.

When we began developing the game, we started off with a GUI that provided us with a display of the process. This was a simple rectangular shape that resembled the to be projected playing field.

Since we did have any form of image processing input at that time, we simulated this by implementing a functionality that allowed us to use the keyboard instead. This allowed us to move a virtual player around the canvas and start working on the desired features, including painting, blending, the change of colours and the selection of different players.

At this stage however, a lot of features differ from those in the final product. The way of drawing for example was first implemented in such a way that the virtual player possessed a 'hand' which it could rotate separately from its own body. This created cone-shaped strokes, rather than the trails players leave behind when walking. Also, the selection of different colours was at first meant to be done by making gestures, instead of walking onto a particular spot on the canvas.

The main reason for these changes was due to the trouble the image-processing part of the game brought along. The difficulty for our game in particular was that we had to focus on the change in movement, rather than just comparing a frame to a static background. We tried many different techniques for the image-processing, so this was a very time consuming part, which ultimately also upheld some other processes from being implemented.

In the end, we came up with a solution that satisfied our needs, but also prevented us from adding some particular features as stated earlier in this chapter.

## 4. Developed functionalities

To make the game working as intended, we implemented a variety of features players can make use of. All of the functionalities are dependant or relying on each other, preserving the simplicity of our game. Still, there are quite some options players can try out.

In order to join the game, people merely have to walk into the sight of the camera and step onto the projected canvas. Once recognized as a player, the particular person will have a colour assigned to him or her, and they'll be able to draw along.

To actually start painting, players have to increase their span, by extending their arms. The larger the created span, the wider the "brush" will become, allowing players to create trails in various widths. This helps players to draw more accurately, but at the same time enables them to cover up larger parts of the canvas at once.

The brush width is not necessarily proportional to the radius the participants make, but rather uses thresholds to maintain stability when painting; small movements and changes in span are easily detected by our system, though this should not hinder players when drawing.

Around the canvas, there is frame, or list, with coloured corners and sides. These coloured spots allows players to change their current colour. In order to do so, players can simply walk on top of them, and continue drawing just the same.

When players enter the game for the first time, walking over any of the coloured areas will immediately assign the particular colour. If participants choose not to, they'll be given the default colour red.

Besides the present primary colours, red, yellow and blue, players also have the option to pick the eraser. This allows players to undo any mistakes by starting off with the colour white - the original "colour" anywhere on the canvas - and repaint the erased spot.

There are more colours to achieve then just the primary ones however. Whenever players paint over an existing trial with a different colour, the trails will blend together, resulting into either green, purple or orange. When all three primary colours are drawn on the same spot, the colour black will appear.

Whenever players are done drawing, they can leave as easily as they'd joined. Walking off the playing field is all they have to do in order to stop painting. The main advantages of this system is that the game itself isn't influenced at all by players leaving; someone else can pick up the work without any problems.

# 5. Interaction design

This chapter will discuss the Interaction Design section, focussing on how we tested the game, and evaluated the results gained from the process.

Applying interaction design principles to our product has been a challenging task. The game has a very steep set-up requirement before it can be played properly; we had to contact many people at different places requesting if there was a possible testing area. This overall lead to many cases which forced us to wait in order to obtain permission. Overcoming these challenges took too much time and most of the possibilities for play testing were missed.

## 5.1 Usability evaluation

To test the quality of our game, we made use of usability evaluation. During this process, we let bystanders observe our game in order to obtain objective information. Furthermore, we let people who weren’t familiar at all with our product play the game, giving us essential feedback about the gameplay itself, and how we could make improvements accordingly.

During the first stage of this process, we made use of empirical evaluation.
When our prototype was developed enough to be playable, we set up our game. Though the setup itself wasn’t complete; we made use of the camera to provide an upper view of the players, and simply used a computer screen instead of a projection, we observed how the game in its current state processed people walking through the canvas.

This allowed us to roughly adjust some thresholds. For example, it became instantly clear that the trail players left behind was far too great in width, so this was something we could change right away.

Furthermore, we approached some of the people and asked if the gameplay itself was clear. Their responses indicated that it was not obvious that players had the ability to change their brush, and neither was changing their current colour.

We also gained some valuable feedback from our project supervisor and teaching assistant. They both agreed that a different method of image processing would create less background noise, which indeed caused a lot of unintended coloured spots on the canvas.

Applying the suggested change heavily improved the overall quality of our game, improving player experience by thus displaying way less bugs.

During the demo of all game context groups, we (finally) were getting close to our ideal setup for our game. Here, we attached a camera to the ceiling, and were able to display the game on a television screen, which was big enough to clearly render the game for the participants.

However, we were still missing the desired projection on the floor, which would give players the full drawing experience. So during the interpretive evaluation process of our product, this was - especially by people who were already familiar with the concept of our game - an often noted improvement.

Perhaps the most suggested enhancement was an indicator for each player, so they were more easily capable of determining their position on the screen. Also, whenever the canvas already contained a large amount of paint, it was very hard for players to see what they were drawing, or if they had any influence on the canvas at all.

Despite the reduced amount of noise resulting from the previous evaluation stage, players were commenting that this could still be slightly improved as well.

Within the last stage of the evaluation process, we made a formative approach to deal with some of the reported problems.

Focusing on the quality of the gameplay, we again made some changes to various of thresholds. For example, the span players have to create by extending their arms has been adjusted to reasonable distance, rather than almost impossible ones.

Furthermore, whenever players first entered the canvas, avoiding any of the coloured sections on the list, they would always get the colour red assigned to them. As for the game in its current state, players will randomly obtain a colour when performing the same actions. This way, we avoided any excess presence of a particular colour.

# 6. Evaluation

In this section we evaluate our product, which includes a failure analysis. As with every product, it is not perfect. It still contains some bugs, which would require attention in a future development process.

## 6.1 Core and rendering layer

In our core layer we have our main game functionalities, such as blending and erasing. In this layer there are no noticeable bugs. We have an high enough test coverage over these classes, to make the assumption that it all works fine.

In our rendering layer we display our game. This consist of the GUI but also the actual drawing of the painting. Just like the core layer, there are no noticeable bugs. Because this layer is mainly GUI the testing is impractical. We have tested it, by running the game and observing what is drawn. As well as checking if the GUI is correct. Even though the test coverage is low, we feel safe to say that it works fine.

In the image processing layer we detect players and convert it to input for the game. This is the most complicated layer and the most error sensitive. It is very dependent on the player, who is unpredictable. We have a mid size test coverage which is mostly due to the use of a lot of OpenCV methods. The bugs in this layer will be discussed in the failure analysis.

## 6,2 Failure Analysis:

- **Double detection**

As of now the image processing doesn't draw circles around the player perfectly. It sometimes draws 2 or even more circles around the player. So instead of just 1 player, 2 or more are drawn. These 'new' players will all get the colour red and can mess up the drawing unless the player is already using red. In that case it is hardly noticeable.

These multiple circles exist due to an interrupted contour. If the contour of the player is interrupted there will be two separate contours instead of just one. Both the contours will have their own circles.

A possible solution for future development would be to check if the newly detected players is too close to an already existing player. So the two new player would both be matched to the existing player.

- **OpenCV crash**

Using Windows can cause OpenCV to crash. It is possible that a java reference exists but the OpenCV object doesn't exist. This error occurs in the OpenCV method *findContours*. There is not much we can do about it. This error has not yet occurred on Mac or Linux.

## 6.3 Process evaluation

The prioritisation of tasks was lacking and the subdivision was near absent. This lead to some blocking elements in the image processing. This also delayed play testing, because all elements needed to be piped together before play testing becomes feasible.

Responsibilities were vastly unclear during most sprints and this did harm the project, if something was done/ready, it wasn't always clear because of this.

Our sprint retrospective while certainly productive ended up being retrospective and retrospective only. No time in was allocated for applying new knowledge of the previous sprint for the next sprint. We did this a bit implicitly but making this more official could have improved further sprints a lot.

Our time estimates for tasks have been inaccurate at times, especially at the start of the project. The estimates did improve near the end, but could have been even better.

## 6.4 Improvements

Subdivision of tasks would have enabled more progress on the image processing side earlier on.

Prioritisation would have moved it higher up on list. Including prioritisation on the sprint plan should have been done earlier. We have learned the value of prioritisation the hard way, but we won't forget it.

Making arrangements for test location even before we were ready to start the testing. This would have avoided the delays later on in the project after we were ready for testing,but weren't able to.

More explicitness about who is responsible for ensuring the task will be completed on time, could have vastly improved our rate of actually finishing all work on the sprint.

Documenting the reasoning for our estimates in the sprint plan, would have helped during reflections. It would help show why a time estimate was inaccurate and increase the chances of more accurate estimations for the next iteration.

## 7. Outlook

In this final section, we will give an outlook regarding possible improvements in the future and the strategy to achieve these improvements. The goal of these improvements may vary from the GUI to the Image Processing component of the game.

As described in earlier sections of this report, we could not implement all the desired features due to various reasons.

Something that we would like to improve is the way of drawing. Leaving a trail when moving is a nice thing on its own, especially with the option to change colours and vary the brush size, but in order to increase the sensation of truly drawing, it would be an improvement to actually draw with your hands instead.

In order to achieve this, we could create a circle around each player, which's colour corresponds to the currently assigned one. Whenever a player wants to draw, they can stick out their hands so that they are no longer within the circle, and start making actual brush sweeping gestures. The width of the stroke increases the farther a players reaches out, so this feature is maintained just the same.

Another feature we would like to improve, which also relates to making use of gestures, is the way of changing the current drawing colour. To do this, we would like players to, for example, hold their hands above their heads. For this we would need to make use of a different technique when it comes to image processing; definitely a real challenge.

The same goes for the addition of opacity when drawing. This would allow players to draw different tints of a colour when they lower or raise their hands regarding the floor, adding more or less pressure to their brush.

A different type of enhancement would be finding a way for OpenCV - the software used for the image processing component -  to smoothly run on the Windows operating system. Now the game really tends to crash after a while, claiming objects become null when they are not supposed to. We did some investigation ourselves, and came to the conclusion that this problem really is operating system based.

A last, and probably biggest change would be adding more types of playing modes to the game. In its current state, the game satisfies many of the requirements we aimed for, adding more modes, the game would become a little less unilateral. The strategy we would apply here is focussing on aspects the game does not cover at all, for example adding a time restraint, or repainting an existing picture.