Creating an Interactive Art Installation for the SmartXP

This thesis describes the process of creating an interactive art installation. A state of the art in both literature on interactive art and interactivity, and in relevant interactive art installations, led to the creation of a Kinect based installation. The different steps of the software that run this installation are described in detail, and the shader code that creates the visuals has been provided. Using this installation a user test was done to find out what people did and did not like about the installation, and what people thought about interactive art in general. The results showed that people want their interactions take effect instantly, and they want to see exactly what influence their actions have on the system. Furthermore a slight preference for strong visuals over strong interactivity was discovered.

Aart Odding
Supervision: Angelika Mader
External Client: Richard Bults
B.S. Creative Technology
University of Twente

Table of Contents

Introduction

The goal of this graduation project is to create and evaluate an interactive art installation. We aim for this installation to be as engaging as possible, such that people playing with it, will not feel as if they are directly controlling the installation, but feel a mutual interaction, with the installation. This document describes the process of creating the interactive art installation, it contains the preliminary research that was done before starting the creation, the usability tests, and a reflection on the entire process.

Interactive art is a type of art distinguishing itself by the possibility of interaction between the viewer, and the artwork itself. It is very hard to provide an exact definition for both interactivity and interactive art, but it is generally accepted that this interaction needs a certain degree of meaningfulness: The fact that a statue or painting can be picked up and placed down upside down, does not make it an interactive artwork, even though the action itself can be considered an interaction with the work.

Before the emergence of the computer, artists that wanted to make interactive art mostly relied on physical manipulation to make changes in the artwork. Marcel Duchamp's Rotary Glass Plates [1] (1920), relied on the participants to manually spin the glass plates. During the 1950's Yacoov Agam started making his Transformable Reliefs [2] compositions that could be rearranged by the audience. His motivation for this was: "to release the creativity of the art public, to encourage people to enter into the spirit of his work and change it according to their tastes".

When computers and consumer electronics became more widely available, interactive art increased in popularity. These new technologies resulted in numerous large changes in interactive art: Motors, actuators and sensors made physical manipulation no longer necessary to influence the artwork. It became possible for the artwork to have a programmed brain of its own. This made it possible for the reactions of the artwork to be influenced by more than one factor, and even previous events. And furthermore computer screens, beamers and digitally controlled lights gave a whole new way of visualising the interactions.

In recent years interactive art has again seen major developments. The rise of the maker movement has popularized the idea of tinkering with code and electronics as a hobby. Not to create commercial applications, but with the intention to play around, and create playful and creative projects. This development has gone paired with initiatives such as Processing [3], and Arduino [4], initiatives that attempt to simplify the use of code and

electronics, to make these technologies more available to non technically educated people and artists. This increased usability of soft- and hardware has led to a new interest in interactive art, this can be observed in the increasing amount of art festivals aimed at digital, interactive, and computer art: ISEA, Ars Electronica, STRP festival and GOGBOT, to name a few.

Research Questions

Because the final goal of this graduation project is to create an interactive art installation, the main research question was chosen to reflect this goal:

Q1 *How to design an interactive installation that is playful, fun and engaging?*

However because this is such a broad and large question to answer the question was subdivided into five subquestions. Each subquestion deals with a specific part or stage of the process as a whole.

First off a state of the art research was done in interactive art. Both in literature, and towards actual installations that have been build. The following research question was used for this process:

S1 *What state of the art in interactive installations?*

When a direction for the installation has been chosen based on the research, multiple impactful choices need to be made on the technical workings of the installation. Human interaction needs to be registered in the installation in one way or another, thus a sensor (or multiple) needs to be picked, the sensor data will have to go through a pipeline of transformations, to process the incoming sensor data, to be usable for gesture analysis.

S2 *How is the interactive installation build from a technical perspective?*

The incoming sensor data needs to go to a processing stage to identify certain gestures of the people interacting with the installation. If for instance the sensor is a distance sensor, then approaching the installation can be seen as a gesture to control certain effects. From this, subquestion three arises:

S3 *What gestures can the installation identify from the incoming sensor data?*

Of course the processing of people's gestures means nothing to them if they are not made visible. What sorts of effects are possible to create, and which work well in combination with the identified gestures?

S4 *What effects can be generated, and coupled to these gestures?*

Finally after the installation is finished it will be evaluated, Usability tests will be held, and factors key to generating interactivity will be attempted to identify.

S5 *What makes the installation inviting to interaction?*

State of the Art – Literature

One of the problems with defining interactiveness is that in a broad way everything can be considered interactive. To play a film one needs to select the "start film" option in the DVD menu, does this make films interactive? Similarly to view a painting one might move around it and look at it from different angles. Does this make paintings interactive? If a painting brings up emotions in the person looking at it, and these emotions in return influence the perception of the painting, does that then make the painting interactive? And if it is in that sense interactive, does this make paintings interactive art? Multiple attempts have been made to solve this ambiguity.

Lopes [5] recognises this problem and suggests two distinctions of interactivity. Weakly interactive and strongly interactive. To explain this distinction he uses the word 'structure' as an analogy for the work of which the interactiveness is being questioned: "In weakly interactive media the user's input determines which structure is accessed or the sequence in which it is accessed, in strongly interactive media we may say that the structure itself is shaped in part by the interactor's choices."

Cornock and Edmonds [6] attempt to solve this ambiguity towards interactivity in a similar manner. However they consider two distinctions too few, and propose a system with four categories of interactive art:
Static: The artwork does not respond to its context. There may be interaction between the interactor and the piece, but this is not visible for observers.
Dynamic-passive: The artwork can change and adapt to variables such as light intensity, temperature, etc. but does not directly interact with humans.
Dynamic-interactive: The same as dynamic passive, but now the changes do no only apply to environmental factors, but also to human interaction with the installation.
Dynamic-interactive (varying): The same as dynamic interactive, but now the installation is also aware of past interactions, which cause the effect of the installation not to be repeatable.

Comparing Cornock and Edmonds' four category system with Lopes' two category system, there is a certain overlap. Lopes' weakly interactive media, is comparable to Cornock and Edmonds' static category, in the sense that neither is intuitively interactive, and is only considered interactive using a loose definition of interactivity. Cornock and Edmonds' dynamic passive category does not fit in Lopes' system. This seems to be because Lopes' system only considers human interaction, and dynamic passive encompasses artworks interactive with non-human variables. Lastly Dynamic-interactive

and Dynamic-interactive (varying) can both be coupled to Lopes' strongly interactive.

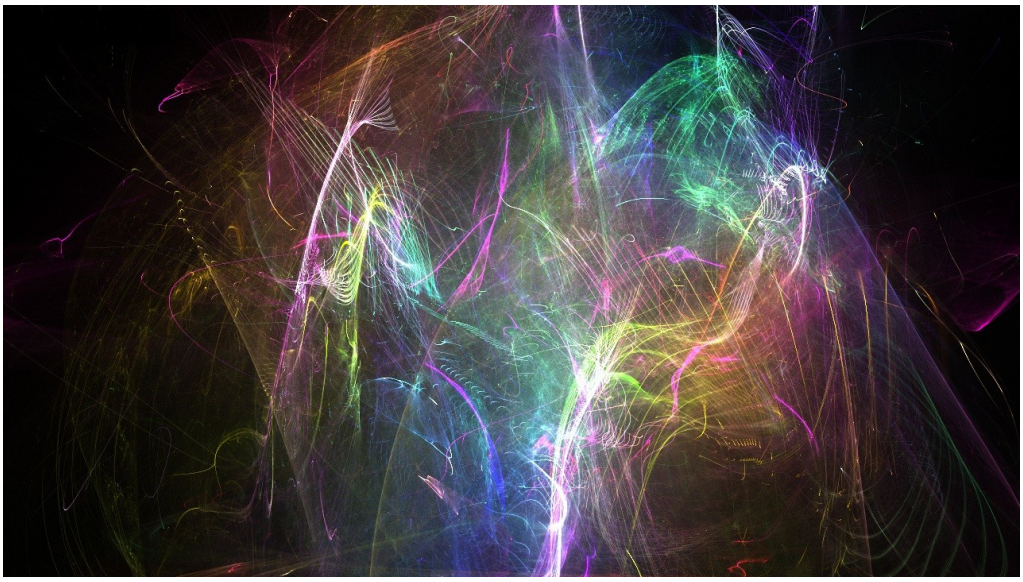
Smuts [7] disagrees with this division of interactivity into multiple categories, and states an action is either interactive or not, but there are no different degrees of interaction. To support this viewpoint Smuts creates his definition of interactivity according to the attributes of social interactions. "Something is interactive if and only if it (1) is responsive, (2) does not completely control, (3) is not completely controlled, and (4) does not respond in a completely random fashion."

The result of this definition has one great advantage, simplicity. Where the other definitions rely on categorization of interactivity in multiple types, Smuts finds a way to define interactiveness that creates a clear boundary between what interactive and not interactive. This is useful, because linguistically it makes more sense. For people not familiar with the subject, interactive art speaks to the imagination, whereas the difference between static interactive and dynamic interactive art is not intuitive. There is however an issue with Smut's definition: A lot of the art that is generally considered interactive art, does not satisfy Smut's rule 3: "it is not completely controlled". Both the artworks by Marcel Duchamp and Yacoov Agam mentioned in the introduction fall in this category, and are thus according to Smuts not interactive.

None of the definitions are completely free of problems, each with their advantages and disadvantages. For this research Smut's definition of interactivity is used as a guideline, this will ensure that the final result is not only responsive, but also active on its own, without people activating it. Furthermore to Comply with Smut's definition it is also necessary to reach the highest category in interactiveness on Cornock and Edmond's scale.

State of the Art – Relevant Work

Beautiful Chaos – Nathan Selikoff, 2013 [13]



Beautiful Chaos works by sensing movement of people's hands using a leap motion sensor. Using this data intricate drawings are produced on a screen in front of the people.

Evaluation

+ Generative: Visuals are generated by mathematical formulas.
+ Simple input (leap motion), still generates sophisticated visual effects.
- Relatively small, only single screen, one person capacity.
- Only one type of visual effects, makes the installation quite monotone.

> "Beautiful Chaos is an experimental art app designed for the Leap Motion Controller, a 3D motion control device that enables gestural interaction with a computer."
>
> "The user interacts with the software and the underlying mathematical formulas to produce an infinite variety of futuristic, abstract shapes. As the user's hands move, the coefficients of a math equation change, and the chaotic cloud of particles ripples and shifts in unpredictable, beautiful and mysterious ways."

Wander through the Crystal Universe – TeamLab, 2016 [8]

This installation build by TeamLab is a room with mirrors on each side, This makes the room look infinitely large. There are lights suspended from the ceiling which the people can move through, creating immersion. Furthermore the people can control the lights with their smartphone, and all the lights are responsive towards each other.

Evaluation

+ Use of space: people are inside the installation.
+ Effects are responsive to peoples location and movements.
+ Fully interactive (Smuts' definition), work not completely controlled.
- App is required for full interaction.
- Installation is based on large amount of hardware.
- Not applicable to this project considering space and budget.

> "This artwork uses an accumulation of light points to create a sculptural body, similar to the way distinct dots of color form an image in a pointillist painting. In Crystal Universe, the particles of light are digitally controlled and change based on the viewer's interactivity with the work. The result is an installation that consists of lights, forming a sculpture that expresses the universe. Viewers are invited to enter and walk around within the three-dimensional light space. This movement affects the light particles and creates changes in the installation. Viewers can also interact with the work by using their smartphones to select elements that make up the Crystal Universe. While Crystal Universe is created by elements selected by the viewers, each action or change affects the other. The viewer's position within the artwork also influences how the work is created; thus, the artwork is continuously changing."

Aether – Thomas Sanchez Lengeling & Gilberto Castro, 2013 [9]

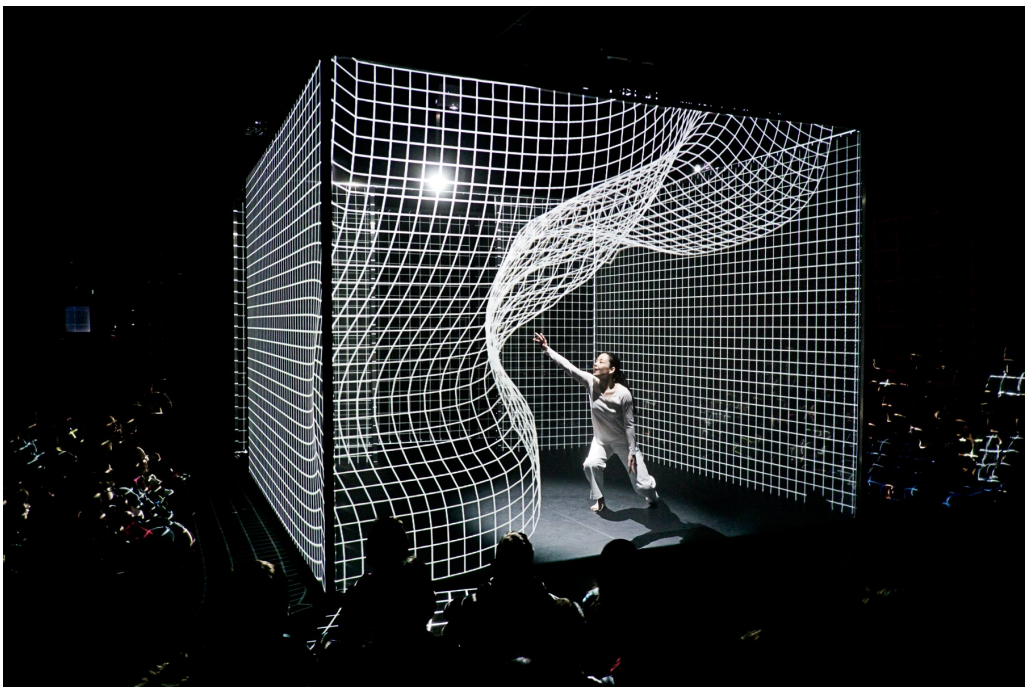
This artwork makes the user feel like they are interacting with a net shaped mesh, with particles continuously spawning and shooting away, creating stunning visuals. The screen is a large touch sensitive plate of glass on which the visuals are projected. Interacting is done by moving your hands over the screen.

Evaluation

+ Beautiful visuals based on Hubble telescope images and geometry.
+ Seems fun to interact with and create visuals.
+ Large scale projection in front of the people.
- Seems unintuitive to touch an artwork.
- Artwork is only responsive to one interaction (touch).

> "A series of studies in geometric symmetry, dynamic particles and interactivity on a large multitouch screen. It explores techniques of manipulating a digital sculpture throughout the interaction of the users. It is inspired by the work of the digital artist Quayola as well as similar inspired modernist cubist works such as Picasso´s "Seated Nude" and the classical French painter Poussin. The users are able to alter the geometry and the physics, which changes the representation of objects from different viewpoints. The sources of study are various Nebula images taken from the Hubble Space Telescope. Those images are exposed into an abstract form, furthermore, is possible to manipulate the digital information in real-time with the touch of the users."

Hakanaï – Claire Bardainne & Adrien Mondot, 2013 [10]



Hakanaï is an installation meant for dance performances. A dancer moves inside of a cubic space on which visuals are projected. The movements of the dancer directly influence the visuals projected on the cube, creating a virtual extension of the dancer.

Evaluation

+ Large space monitored.
+ Not flat, but 3d because of visuals on all four sides.
+ Reactivity between body movements and the visuals.
- Only one person at a time can control the installation.
- Mostly meant as watchable performance, in which case it is not interactive for the crowd.

> "Dance choreography performed in the immersive environment of a moving cube, to explore the fleeting nature of dreams and the fugacity of life."

> "Live animations based on physical movement modelling, on an original music score played live. After the quadrifrontal performance, the audience will be invited to explore the stage installation."

Drawing on the Water Surface – TeamLab, 2016 [11]

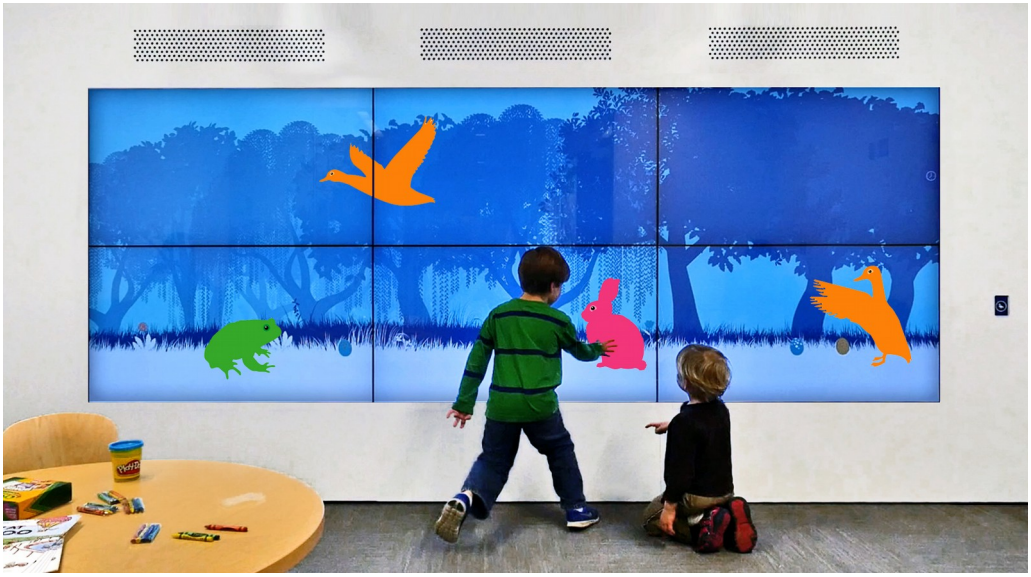
Another installation by TeamLab: Koi (carps) are projected onto a layer of water. People can enter the water to play with the fish and see the fish play with them.

Evaluation

+ Interesting play with position in space deciding outcome of visuals.
+ Original idea to make projections on water.
- Visual effects are rather basic and monotone.
- Very impractical to fill a room with water.
- People have to take of their shoes and get their feet wet.

> "Koi swim on the surface of water that stretches out into infinity. People can walk into the water. The movement of the koi is influenced by the presence of people in the water and also other koi. When the fish collide with people they turn into flowers and scatter. The trajectory of the koi is determined by the presence of people and these trajectories trace lines on the surface of the water. The work is rendered in real time by a computer program. It is neither a prerecorded animation nor on loop. The interaction between the viewer and the installation causes continuous change in the artwork. Previous visual states can never be replicated, and will never reoccur."

Forest Friends – potion design, 2015 [12]



Forest friends is an installation that helps children deal with the mental stress of going through treatment in hospital. Each child has an animal as an avatar, that accompanies them in the hospital, and helps them in the sometimes harsh process of treatment.

Evaluation

+ Good looking visuals.
+ Keeps track of people over time, so it is possible to come back and continue play.
- Requires unique RFID tags for everyone.
- Only fun if people keep coming back to it.
- Less suitable for public installation where many people walk by.

> "The digital woodland welcomes patients and their families in the Waiting Room. Thereafter, patients trigger their animal companion (frog, duck, bunny) by scanning their hospital bracelet."

> "The patient's animal accompanies them through all stages of treatment and, when cued, demos behaviours like sleep during anaesthetic administration or departure at the end of a visit or wait."

Conclusions and Ideation

When comparing the choices made by the other large projects mentioned before, combined with our initial idea and vision, we can come towards multiple conclusions as to what the installation should look like and how it should work.

First and foremost there are certain criteria to the contents of the installation in order to qualify as an "interactive installation". As concluded before Smuts' definition of interactivity is used as a criterion for interactiveness. This means that the installation responds to human activity, shall not be fully controlled, shall not fully control, or be completely random. This means that things like playing a video after certain triggers are not interactive enough.

The output of the installation will therefore be computer generated graphics, influenced and manipulated by the people using the installation, and projected onto a wall in front of the people like in Hakanaï and Beautiful Chaos. The visuals will be projected onto a wall or screen in front of the people, which will hopefully create a very direct connection between the people playing, and the virtual interactions that they control.

The visual style of the installation will be kept simple but colourful, while containing lots of movement, in order to create a very dynamic visualisation. Furthermore some of the effects will be kept rather abstract, instead of a very direct representation of the people's features or movements, to create some confusion and mystery to add to the wow factor. A Moodboard has been made portraying the intended style: figure 2.

Like all the works mentioned before, apart from Hakanaï, and Beautiful Chaos, we chose to have the requirement that multiple people should be able to interact with the installation at once. This so that the installation is not only and interactive experience, but a social experience as well. This requirement already eliminates technologies such as augmented and virtual reality and any type of body mounted sensors. However this is not a problem, because it is also better for the installation if people do not need extra equipment to be able to interact with it, because such equipment would quickly increase the threshold for people to start playing with the installation.

Therefore Microsoft's Kinect sensor was chosen to take that role. The Kinect is a camera that senses depth instead of colour, which makes it able to detect people standing in front of it. This allows for very natural interaction because the participant does not need to physically manipulate some device to

initiate interaction, but can freely move through the space in the installation to interact with it. Furthermore because the data that comes from the sensor contains so much information, many different variables can be used to facilitate interaction, movement, moving ones arms, jumping, ducking, touching other people, making your body larger and smaller, and many more gestures can be deduced from the data send by the Kinect.

One issue the Kinect has is the limited field of view. Because the idea was to create a very large area as a playground for the interactions, and a life size projection as output, it was decided to use multiple Kinects standing next to each other as input. This however means that the data coming from multiple Kinects has to be combined in such a way that the whole play area is being sensed as if it were through one sensor. Some parts of the area will be covered by two Kinects, while other parts might not be covered at all. This will need to be compensated for in the software. A general impression of the whole installation can be found in figure 1.

Figure 1: Impression of what the finished installation could look like.

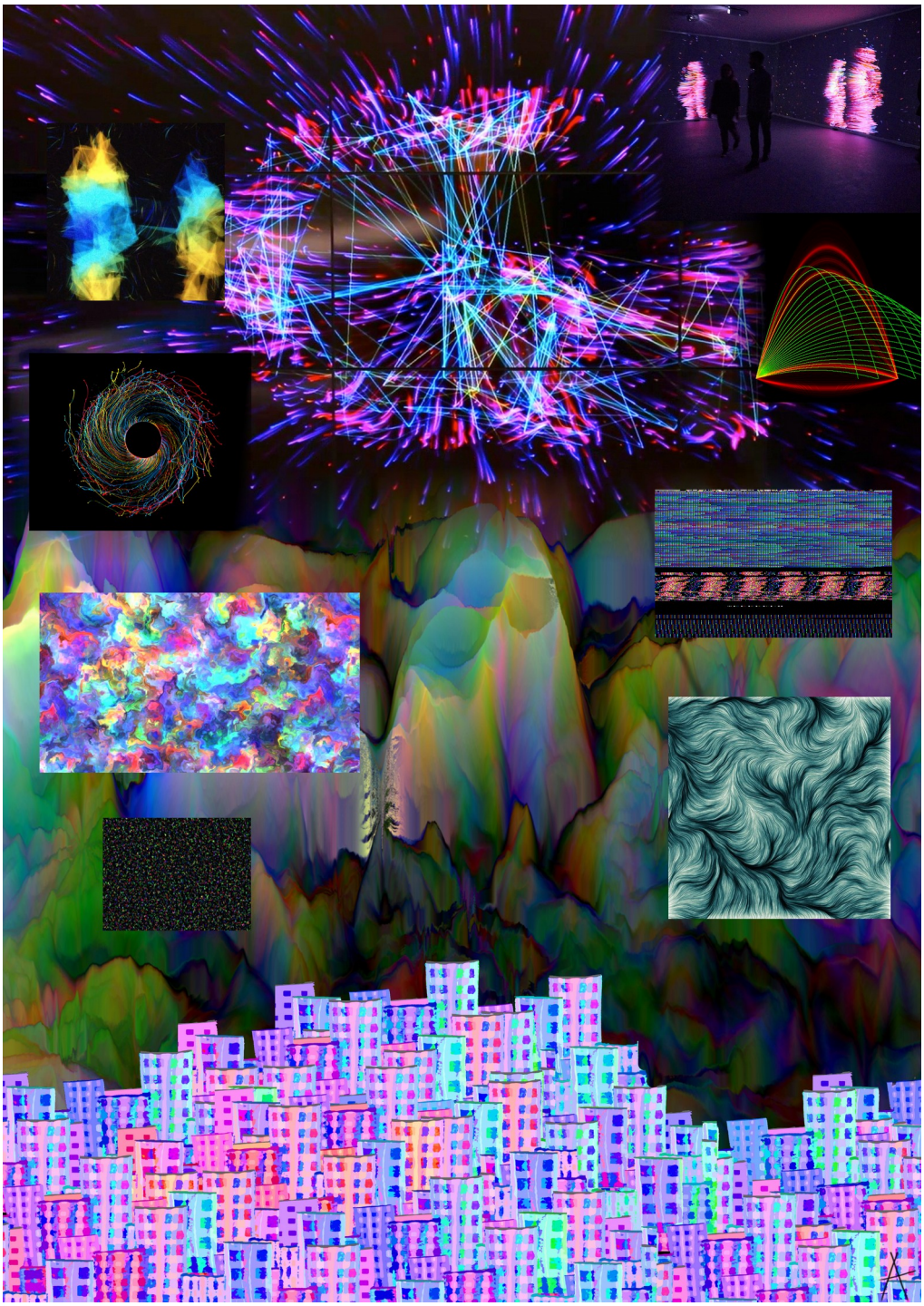


Figure 2: The moodboard, functioning as a visual style guideline for the installation.

Requirements

Following the ideation phase a set of requirements was devised. These requirements were made with two goals in mind: To ensure that a certain set of goals which has been described in the ideation phase will be met. And secondly: To be able to look back at the requirements when the projects is finished, as a measurable benchmark, and to see how well the creation process went.

Interaction requirements

- Interaction must somewhat change over time, so the installation stays unpredictable and replayable.
- People must realize how their interactions affect the visuals.
- The interactions must be understandable, E.G. if a certain effect is triggered when a person does a somersault no one will ever find the effect.
- Interactions should be doable for all people. Children and grown ups.

Functional requirements

- Must support multiple people interacting simultaneously.
- Must support output of images on multiple screens/ beamers.
- Must support the combination of the image of multiple Kinects into a single input.
- Must have a sufficiently large screen so that people can see themselves at full size.
- Must be efficient enough to run at Kinects full speed, without hiccups.

Visual requirements

- Must contain certain visual complexity, E.G. Not just some coloured rectangles that move.
- Good use of colour: Too much would make the installation look tacky, while too little would make the installation look dull.
- Must also look good when there are no people interacting with the installation.
- Visuals should at least somewhat change over time, to allow for replayability.
- Must look inviting for people to interact with, and attract people to interact with it.

Realisation – Combining multiple Kinects

For people to be able to interact with the installation, it needs a method through which it can sense participants' gestures. The installation can then give responses to these gestures to stimulate interaction and facilitate play. For this installation Microsoft's Kinect sensor was chosen to take that role. The Kinect is a camera that senses depth instead of colour, which makes it suitable to detect (shapes of) people standing in front of it. This allows for very natural interaction because the participant does not need to physically manipulate some device to initiate interaction. Furthermore because the data that comes from the sensor contains so much information, many different variables can be used to facilitate interaction, movement, moving ones arms, jumping, ducking, touching other people, making your body larger and smaller, and many more variables can all be deduced from the data send by the Kinect. However this comes at the cost of the necessity of more complicated code analysing the sensor data to extract these action. Another issue with the Kinect is the small field of view of approximately 58°, this means that even though the Kinect is the ideal sensor to to track the motion of people through a space, this space can only be relatively small. To counteract this problem it was decided to use multiple Kinects next to each other to increase the effective area of the sensor. Unfortunately this again comes at the cost of increased code complexity. The following chapter will explain the process that the Kinect data goes through, to go from raw depth values to usable information. The first part will describe the process that is run for each Kinect individually, while the second part will discuss the process that is run for all the Kinects combined. An overview of the process that the data goes through from Kinect to visuals can be seen in figure 3.

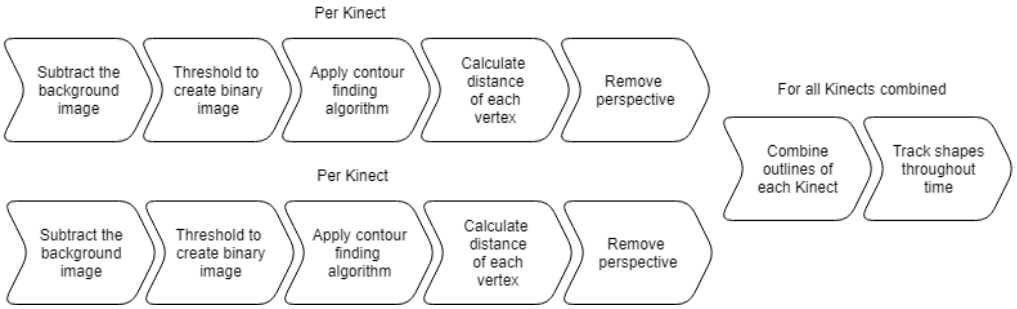


Figure 3: The pipeline of transformations data goes through before usage for visual effects.

Input

To interface with the Kinect over USB a driver is required. For this project libfreenect [14] was chosen for its minimalistic design, and cross platform capability. Libfreenect provides the developer with a 640 * 480 array of shorts (16 bit integers), at a constant 30 frames per second. These shorts contain a depth measurement in millimetres at that specific pixel. These measurements can range from approximately 500 mm (50 cm) to 4500 mm (4.5 m). However sometimes sensor read errors cause the pixels to be either 0 or 10000.

Because further processing relies on external libraries that do not use this uncommon pixel format, the data is first converted to the more common one byte per pixel, single channel pixel data. This format is most commonly used for black and white bitmap images, and thus supported by most image processing libraries. Because this conversion between pixel formats needs to be done extremely often, up to 37 million times per second (running 4 Kinects at 640p 30 fps), it is important to be efficient as possible.

A naive implementation could be a clamped map function, which would look something like this:

```
if (pixel_value < 500)
{
    result = 0;
}
else if (pixel_value > 4500)
{
    result = 255;
}
else
{
    result = (255.0f / 4000) * (pixel_value – 500);
}
```

However this can be very expensive when executed so often: two branch conditions, a possible division, and multiple float to integer conversions.

A better approach is to use a lookup table so that no calculations are necessary. The difficulty here stems from the fact that the possible pixel values range from 0 to 10.000 and making a lookup table of 10.000 elements wouldn't be very fast either because that much data does not fit in cache simultaneously.

Furthermore the only real values that are important are between 500 and 4500 (the sensing distance of the Kinect). Coincidentally this gives a range of 4000 values which is very close to 4096 which is 2^12. This is useful because we need to convert to a range containing 256 values (0 to 255), which is 2^8. This means that when we bitshift to the right by 4 bits we naturally map the 4096 range to the 256 range.

Unfortunately it isn't as easy as adding 500 and bitshifting right by 4, because there are still the special cases of 0 and 10.000. 10.000 + 500 is bitshifted right by 4 equals 656, this means that a lookup table to 656 elements would suffice to handle all cases, 656 bytes is small enough to fit in cache at once, because a lookup table is used no branch conditions are necessary, and no more information is lost than in the naive implementation shown before. The code could look like this:

```
result = lookup_table[(pixel_value – 500) >> 4];
```

Now that the images are stored in a more conventional format, they can be viewed as a regular black and white image: figure 4.

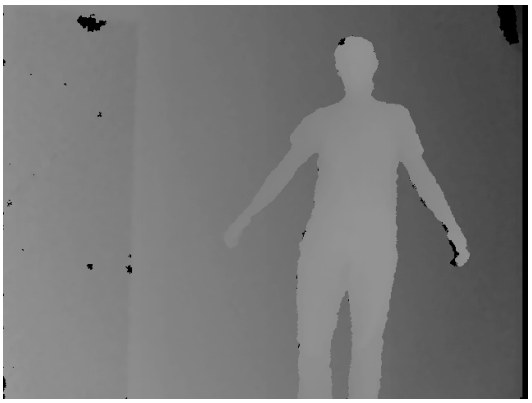


Figure 4: Example of incoming Kinect data mapped to brightness values.

Figure 5: The background of the scene in figure 4.

Removing the background

Even though as a human it is very simple to spot the human in an image, this is much harder for a computer. One effective and not to complicated method involves subtracting the background of a scene from the image (assuming the background does not change too fast) in order to only be left with the people standing in the foreground. In the case of figure 4, the background that should be subtracted can be seen in figure 5.

To subtract the background the background first needs to be known. The background is the image the Kinect sees, when there are no people standing in front of it. To know when there are people in front of the Kinect a PIR-sensor (infrared movement sensor) is used. Now when there are no people in front of the Kinect the application can use the incoming images to sample the background. This also allows for slow changes in the background, without it being problematic for the next stages.

Three tricks are applied to keep the background more accurate. First, when a sample is being taken of the scene not one frame is used but around 30 (1 second worth of frames). These frames are not averaged, but the maximum value is picked at each pixel. This is done so that the edges of things in the scene becomes less fuzzy. Secondly, the background does not consist of one sample, but each time a new sample is taken it is added with a certain weight to the previous samples. This makes every older sample have less impact on the background, while the background is a still bases on multiple samples. Lastly samples are not taken continually, but they have a timeout. This means that the background always reflects the background during a larger timeframe, instead of in a single moment. The combination of these factors allows for excellent background subtraction: figure 6&7.

Figure 6: The leftover image after the background is subtracted from the input.

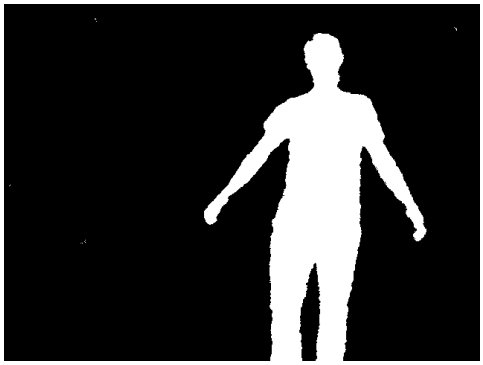


Figure 7: The subtracted image after applying a threshold.

Thresholding and searching for contours

Even though the computer can now differentiate between people and background the data is still not very useful because it is in image (bitmap) format. To get convert between bitmap data, and vector data, OpenCV [15] has a useful feature called contour finding, which does exactly this. It takes a binary image, and finds the contours of shapes in the image. OpenCV also contains functionality to convert the byte image to a binary image using thresholding (figure 7). The result of the contour finding algorithm can be seen in figure 8, and in figure 9 after applying boost.geometry's [16] simplify algorithm.

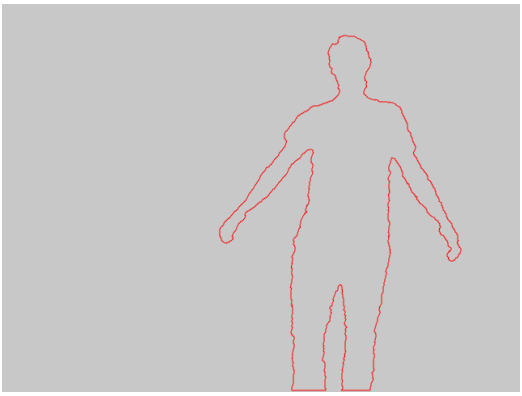


Figure 8: The vector data obtained by OpenCV's FindContours.

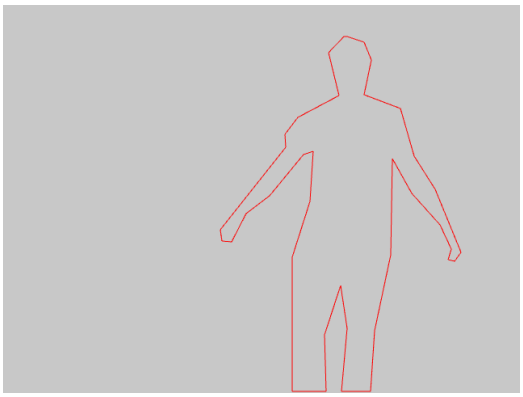


Figure 9: The remaining shape after applying boost.geometry's simplify.

For the next steps some clarification is first necessary. Because the final goal is to use multiple Kinects together, instead of just one at a time, it is necessary to map the vertices that have been found to one shared coordinate system between all the Kinects. To realise this the vertices should first be transformed to a 3D coordinate system relative to the Kinect, after which the multiple Kinects can be added together by some transformations.

Right now the coordinates stored in the vertices found by OpenCV are the indices of the pixels that consisted the shapes in the images. This means that the vertical position relative to the Kinect cannot directly be determined from the vertical position of the vertex found by OpenCV alone. The real location is also dependant on the distance to the Kinect. Fortunately the Kinect is made for just that: being a depth camera.

Calculating vertex distance

The x and y values stored in the vertices of the contours now correspond to the x and y indices of the pixel in the image. Using these indices it is possible to sample the depth at said point from the original depth data received from the Kinect. The problem with this is that each vertex in the contour falls exactly on the edge of the contour in the picture. This means that half the times the depth is sampled on a certain vertex, this position actually falls outside of the contour in the depth image of the Kinect, which would yield inaccurate depth values.

To counter this problem it is necessary to measure the depth not exactly on the vertex position, but to move the sample position slightly inwards in the shape of the person, as seen in figure 10. This is possible because the contours found by OpenCV always have a clockwise winding. This means that the inward normals at each vertex can be found with some trigonometry, after which a new sampling location can be found slightly inward in the shape, along this normal.

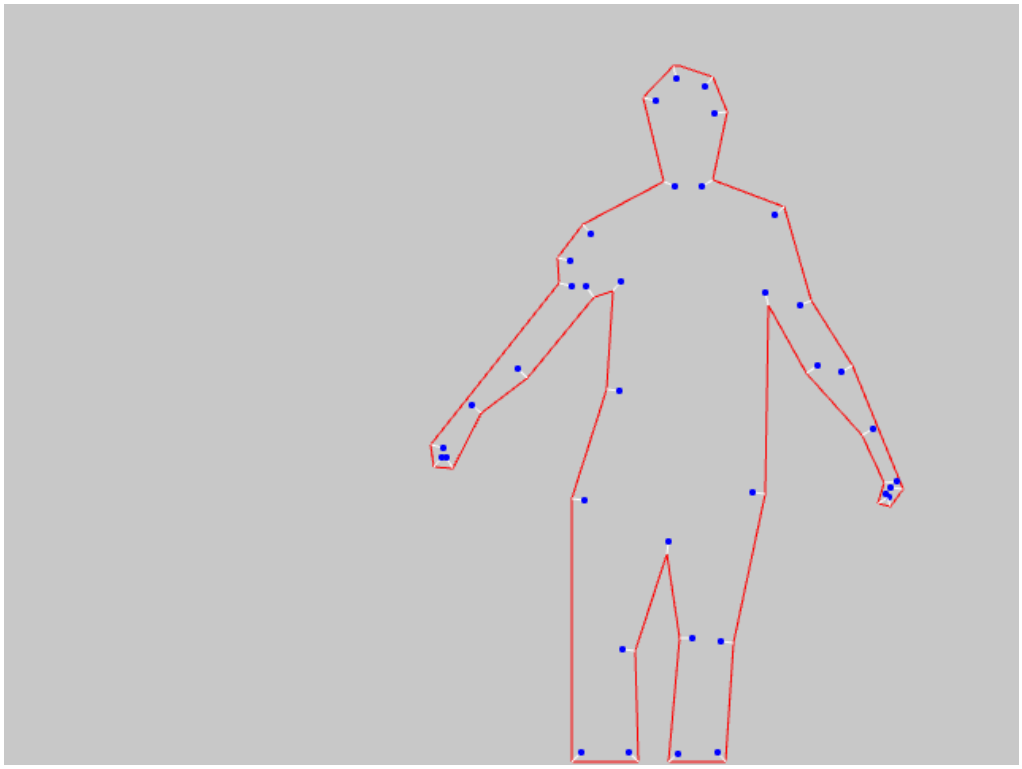


Figure 10: The sampling locations in blue, sampling the depth of the vertices pointed to in white.

Even though this technique of sampling improves the amount of erroneous samples tremendously, there are still occurrences of samples outside of the contours in places like fingers, and Kinect interference errors that cause false depth values. Because of this it is also necessary to filter all the depth values, to take out the false readings.

To solve this problem each sample goes through a validating stage, to make sure there are no invalid samples. If a sample is found to be invalid its distance will calculated by averaging the two nearest valid neighbours.

It is first important to identify what types of error can make a depth reading inaccurate. There appear to be three types: A. Interference from infrared light, either from other Kinects or sunlight. B. Taking the sample outside of the body, which causes the depth to be the depth of the whatever is behind that person. C. Taking a sample in the infrared shadow that every object in front of the Kinect creates. This shadow is a side effect of the fact that the infrared laser, that the Kinect uses to sense depth, is not on the exact same spot as the infrared camera. To gain a better idea of the nature of the false depth readings the readings of a single figure were graphed: Figure 11.

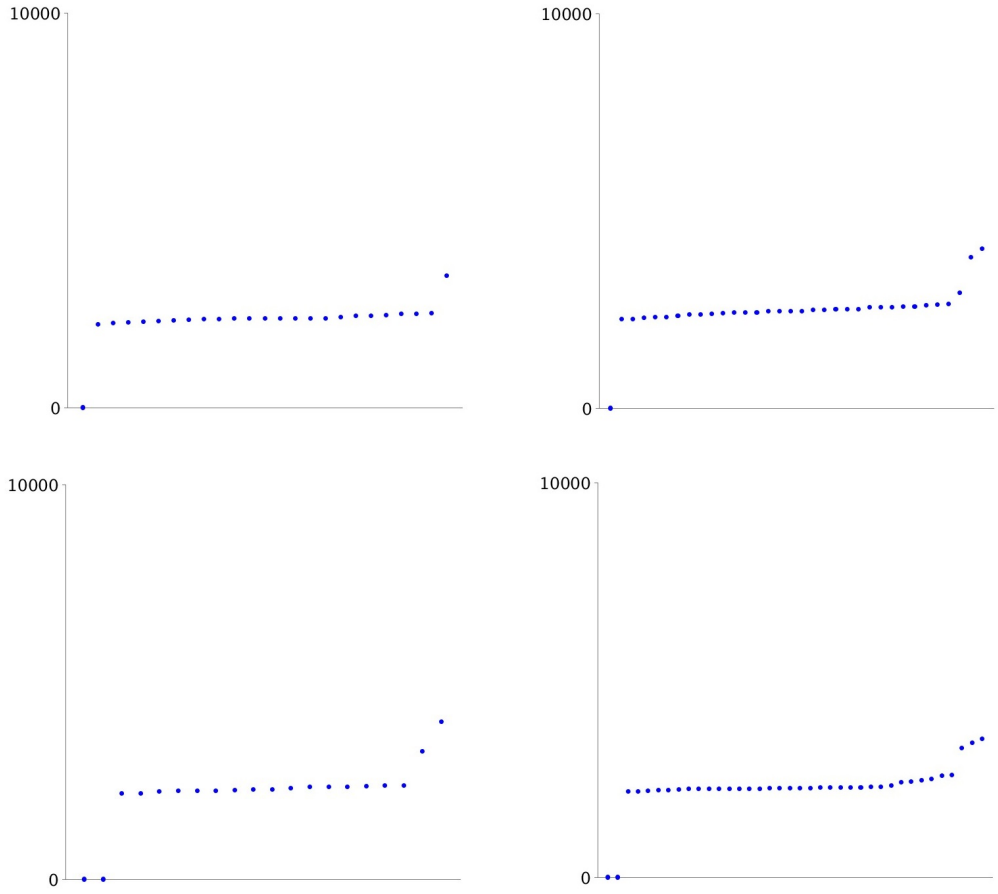


Figure 11: The depth readings of four people as seen by the Kinect visualized: values sorted by distance (mm).

Of these errors, C is very easy to solve: invalidate all depth readings of 0 (total blackness). A and B are a bit more subtle, The interference appears to always to too bright, however its brightness is not consistent, and measuring outside of a person's body gives to dark values, but this is also dependant on what is behind the person. When analysing the distribution of the depth samples from a single person standing in front of the Kinect, the erroneous depth values become apparent. There are three distinct groups: Samples that are to near (A & C), samples that are valid, and samples that are too far (B). A good approach towards filtering the samples seems to be to sort them first so that the median can be obtained, then a range centred on the median can be defined as valid. For each depth measurement that is found that is not valid, the average of the two nearest valid measurements can be taken.

Once the depth has been established for each vertex of a person, it is also possible to change the x and y coordinates of the vertex from being locations in an image to locations of the world relative to the Kinect. The first step is calculating the dimensions of the Kinect's vision at the depth of the vertex. The width at any distance can be found with: *depth (m) * 1.08*, and the height at any distance can be found with: *depth (m) * 0.825*. Then the relative position of the vertex in the Kinect's depth image can be mapped to the same relative position in the dimensions calculated before, in order to find the 3D coordinate of a vertex relative to the Kinect. Because these coordinates no longer contain perspective, they can now be combined with the data coming from other Kinects.

For all Kinects combined

Combining Kinect shapes

To combine the incoming data of multiple Kinects, it is necessary to know the relative positions of the Kinects to each other. The vertex data of each individual Kinect is relative to the Kinect itself, thus a different linear transformation is necessary to transform the data from Kinect space to shared world space.

Now that all the coordinates are in the same coordinate space, the data of a person that appears in two Kinects overlaps, as can be seen on the left side in figure 12. It is now necessary to reduce the two outlines of a person together, so that each person has only one outline, independent of in how many Kinects they appear.

There are two ways to this: Using a geometry library to find the spatial set theoretic union of the two outlines, or drawing the outlines on a flat surface, and then reuse OpenCV to find contours of the combined outlines, the disadvantage of this technique is that depth information is lost.

Comparing these two methods, the first option seems the superior, it is both more efficient and preserves more information. The problem with this approach however is that the geometry library that has been used for simplification as well: boost.geometry, has problems with self intersections in the outlines, created when the perspective is being removed from the outlines. Thus it was chosen to use the drawing method.

First a framebuffer object (FBO) is created using OpenGL. An FBO is an image that can be drawn onto as if it were a regular OpenGL window. The advantage however is that the pixel data can easily be pulled back to the main memory after the drawing has finished. This means we can apply the OpenCV contour finding algorithm on the pixel data, to retrieve all the combined outlines.

Another simplification is necessary, because OpenCV considers each pixel a vertex, which creates much too large vertex sets. Boost.Geometry is used for this simplification. The output can be seen on the right side in Figure 12.

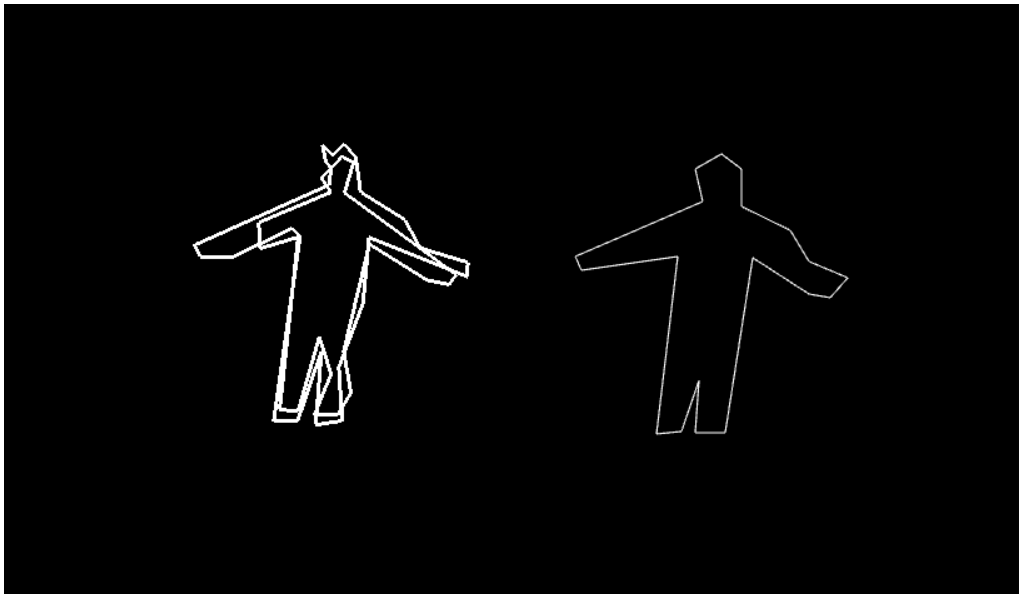


Figure 12: The outlines of the same person in two Kinect, before and after combination.

Tracking shapes over time

The end goal of the installation will be to couple people's gestures to visual effects in the installation to create interactivity. To identify gestures, it is necessary to have knowledge of people as entities over time, instead of a collection of people's outlines, without temporal continuity.

Between any frame N, and its next frame N+1, people can do three basic things: enter the sensor area, leave the sensor area and move. However because the Kinects perceive the space in front of them as a 2D image instead of a 3D space, two extra actions are possible: People walking past each other, can together become one outline: Merge, and afterwards separate again: Split. This gives a total of five possible actions: "Move", "Enter", "Leave", "Merge" and "Split".

To determine which action an outline undertook and in which (if any) outline that resulted, it is first necessary to look in broader terms: which outlines, between frame N and N+1, have any relation at all. A person in the left side of a room in frame N, cannot suddenly be on the right side of that room in frame N+1. Thus relatedness is dependant on position. For simplicity we consider two outlines related if they intersect.

For each outline in frame N+1 a list of intersections with outlines in frame N is created, and in reverse for each outline in frame N, a list of intersections with outlines in frame N+1 is created. Note that these two lists do not need to be symmetrical, because of splitting and merging of outlines. These intersections are the relations between the outlines in both frames, however they still need to be categorized into one of the five aforementioned categories to hold any meaning for the flow of people moving in and out of the installation.

To categorize the relationships can be done with counting. If outline A has a single intersection: outline B, and B also only has a single intersection: A, then A and B are the same person, and their relationship is Move.

If outline A has a single intersection: outline B, but B has two intersections: A and C, then the relationship is either Merge or Split: Merge if A and C are in frame N and B in frame N+1, and Split if B is in frame N and A and C in frame N+1.

All the outlines that have not yet been assigned are either categorizable as Enter or Leave. Enter if the outline is in the new frame, and Leave if the outline is in the previous frame. From this information outlines can be assigned to Person objects, which can be analysed for gestures over time.

Realisation – Gestures

First attempt

The interaction in the installation comes from the coupling of certain identified gestures to effects. IE. a person jumps, and some virtual objects that are part of the effect also move up. To create this sort of interaction, the possible gestures that were deemed identifiable, were determined: they are listed below.

Jumping: Can be recognised by a rapid increase of the y-coordinate of the lowest point in the body.

Ducking: Can be recognised by a rapid decrease of the y-coordinate of the highest point in the body.

Spreading arms: Can be recognised by the increasing distance between the maximum and minimum x-coordinate in the body.

Pulling in arms: Can be recognised by the decreasing distance between the maximum and minimum x-coordinate in the body.

Movement: Can be recognised by the displacement of the centre of mass of the body.

Arm-Tracking: The idea of tracking the arms was also explored, multiple possible ways of arm-tracking where deemed good enough to be usable, they are listed here:

The first method tried to find the direction of the majority of extreme points relative to the centre of mass of the body. This is done by first translating the body-outline such that its centre of mass lies on the origin of the coordinate system. Then by squaring the magnitude of the vectors comprising the body-outline, vectors that already have a large magnitude have now become even more extreme. This means that the new centre of mass of the outline moves in the direction of the most extreme points. When taking the direction from the old centre of mass to the new centre of mass, the direction of the most extreme points is found. If a person sticks out their arm then this will be that direction. One caveat is legs often stick out more than arms, which gives inaccurate results. A very easy way to solve this problem is to assume the people are standing upright, and just ignore all the points that lie below the centre of mass.

The second method uses the Ramer, Douglas-Peucker[17, 18] recursive simplification algorithm. The algorithm simplifies until a given error is met, this means that using the right error (the thickness of a hand), hands will always be reduced to a single point, while body parts thicker than a hand (such as the head), will keep more than one point. Because the arms are relatively long, compared to their thickness, the angle of the two line segments at the hand point will be very sharp. Arms can now be detected by looking for corners sharper than a certain threshold. The algorithm is provided by the Boost.Geometry library.

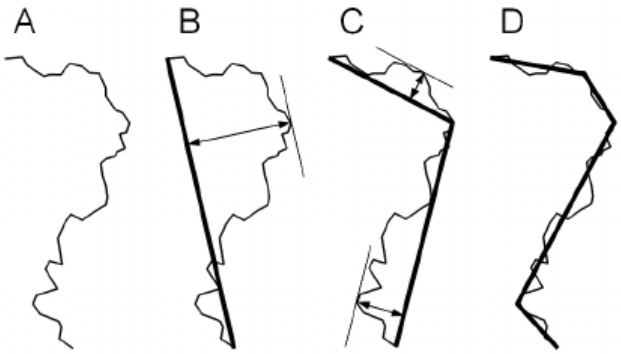


Figure 13: Polygon simplification using Ramer, Douglas-Peucker.

The third method is somewhat different in its approach, in the fact that it does not attempt to just identify the arms, but the general skeleton inside of the body. One way to do this is to find the topological skeleton/ medial axis of a polygon, as defined by Blum [19], this defined as the set of all points having more than one closest point on the object's boundary. Unfortunately there are no open source library implementations of the medial axis problem. However a very similar problem: the straight skeleton problem, Aichholzer [20], does have an implementation in the CGAL computational geometry library. The straight skeleton is mainly different from the medial axis in the fact that it only contains straight lines, where the medial axis can also contain parabolas, as can be seen in figure 14.

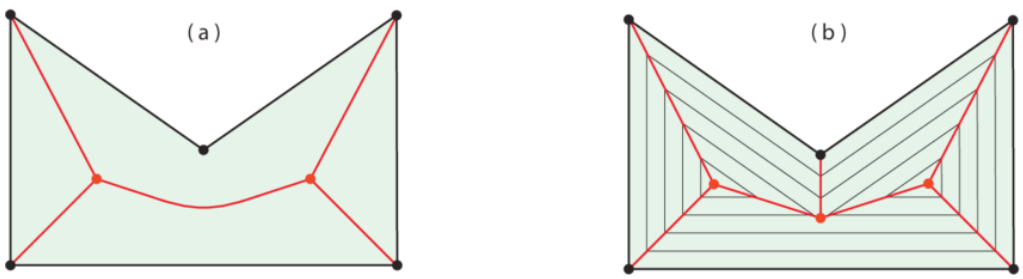


Figure 14: Comparison of Medial Axis (a) and Straight Skeleton (b) of the same polygon.

Unfortunately when using this algorithm on the outlines found by the Kinect, it slowed down the whole application too much, going from never dropping below 60 frames per second, to reaching an average of five frames per second.

The last alternative is to use a Voronoi diagram creation algorithm and subsequently filtering out the edges that do not fully lie withing the body-outline. An example of this can be seen in figure 15.

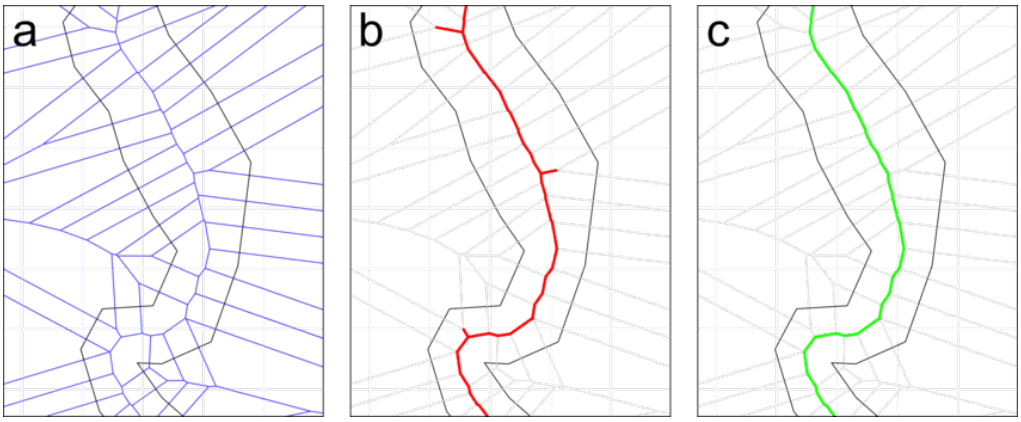


Figure 15: Filtering the Voronoi diagram.

With this approach the creation of the Voronoi diagram was relatively fast, and did not slow down the application to unusable speeds. However the second filtering stage, is very costly. One way of filtering is to use a point inside polygon algorithm to check for each line segment if one of the two points falls inside the body-outline. However even assuming the fastest of these algorithms runs in O(n) time, then to check all the points it would still run in O(n^2). This proved to not be fast enough to run in real time.

Evaluation

Most gestures that were considered identifiable where relatively easy to implement, and ran very fast. However the arm recognition proved somewhat of a trouble. The first two methods (direction of most extreme point, and simplification) worked without too many concerns, however they would sometimes glitch out, or not recognize features. The third method (straight skeleton) looked very promising, but was hard to make performant enough to run in real time. Although possible to create to Voronoi diagram with relative ease. It took too much cpu time to actually filter all the line segments. This problem could have been solved by parallelization, or hardware acceleration (using texture lookup), since all the filtering calculations are independent of each other, however in the end this was not considered worth the effort due to larger problems with the approach of gesture identification in general:

The initial idea was to identify gestures, and then couple certain events in the effects to these gestures, as a means of creating interaction. However when the first effects were being made it quickly appeared that this does not feel truly interactive. As Smuts said: “Something is interactive if and only if it (1) is responsive, (2) does not completely control, (3) is not completely controlled, and (4) does not respond in a completely random fashion”. In this case we can speak of complete control, which means the whole installation wouldn’t even be considered interactive at all. The moment a person realizes the effect a certain gesture has on the visualization the magic is gone. Being in complete control does not stay fun for long. Therefore a new less direct way of gesture to effect coupling was explored.

Second attempt

The problem with the first iteration of gesture recognition and usage comes from the full control the person has over the installation. Full control is predictive and thus boring. A better way of creating interaction is to first start with the effect. The effect should be running continuously, autonomously, and unpredictably, and people should only be able to influence the way the effect runs, while staying unpredictable. There is a big difference in knowing something will change and knowing exactly what will change.

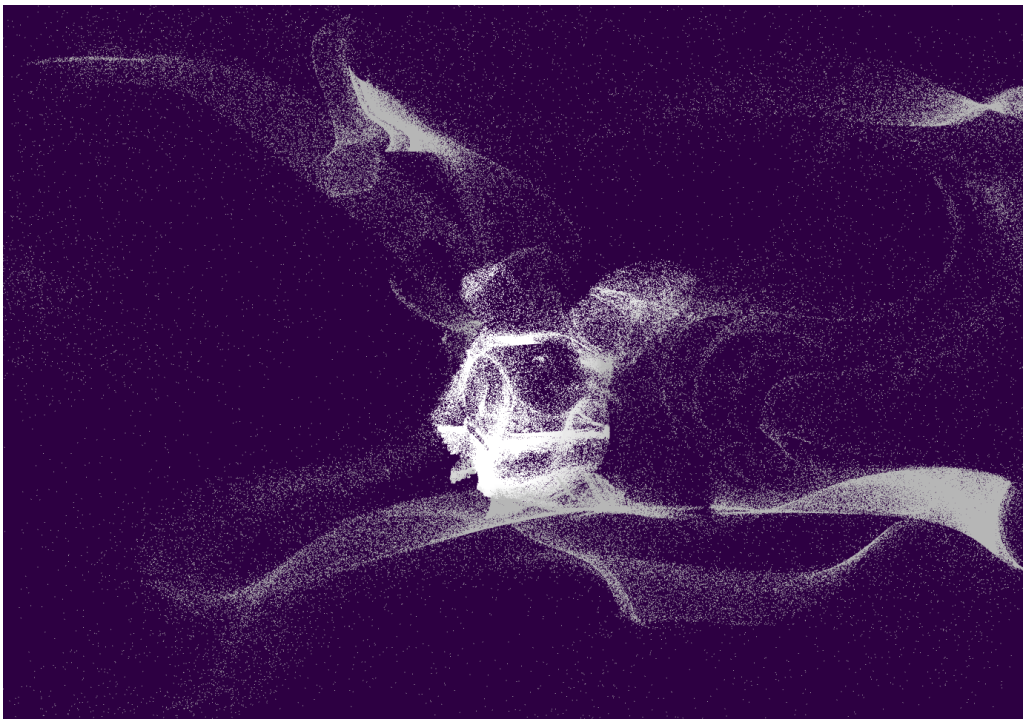
Evaluation

This idea was initially tested by creating a large particle system, moving with to simplex noise (derivative of Perlin noise). Instead of directly controlling the particles with gestures. The people interacting can only somewhat adjust the particles that they touch, to make them move in different patterns. This method proved to be much more successful in creating interactivity and was therefor much preferred over the direct control method, which caused a certain design pattern that was eventually used for all the developed effects: First create a system with its own behaviour, and only afterwards introduce a variable that people can influence by their interactions.

Realisation – Effects

The final installation was finished with four separate and unique interactive effects. To create the effects inspiration was taken from the moodboard created: Figure 2. The shaders that were used for each of the effects can be found back in Appendix C.

Effect A: Particle system in flow field

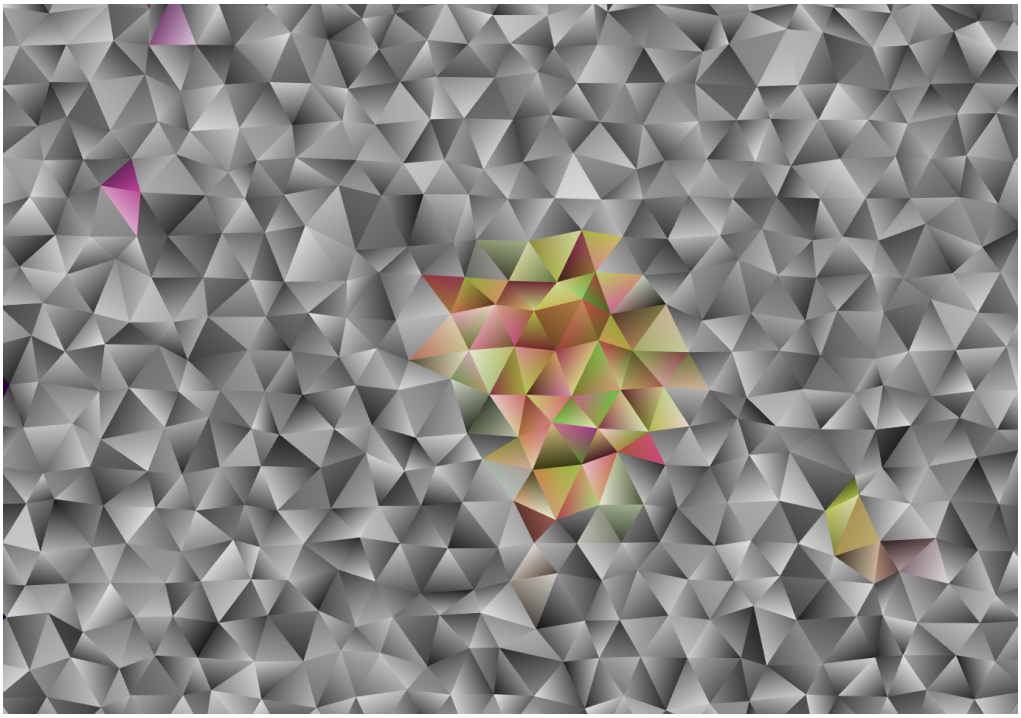


When looking at other interactive art installations the theme of particle effects is often used. From quite early on there was the idea to make a particle effect where each particle takes its direction from a Perlin noise field. The movement was programmed in GLSL (OpenGL shader language) to provide the parallel processing power of the graphics card. Furthermore the noise implementation used was simplex noise 3D by Ashima Arts and Stefan Gustavson.

To create the interaction it was first attempted to either only show the particles on the location of the screen where people are standing, or to remove the particles where people are standing. Neither gave the desired result of interacting with the particle system, but only gave the person the choice where the particle system was visible. In this case the particle system could just as well have been a video.

The answer was to only show a small part of the particles in the system, and revealing all of them when a person touches them. This created both interaction and an incentive to play with the installation.

Effect B: The Shattered mirror

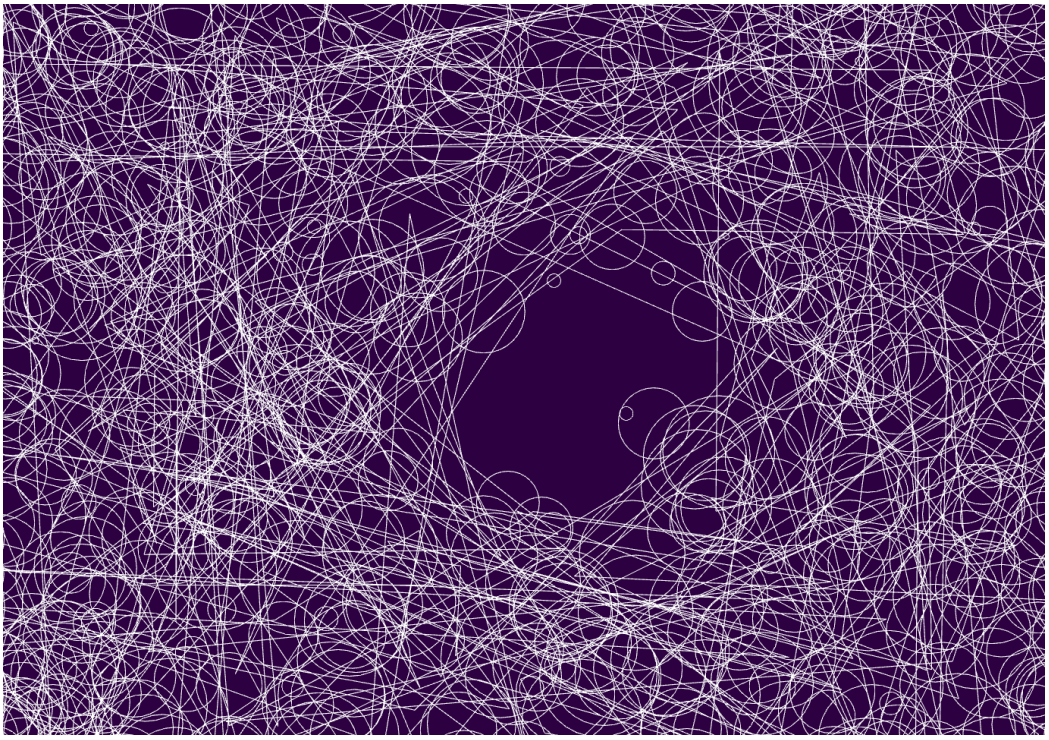


For the second effect the initial idea came from John Conway's Game of Life. A "game" where cells in a grid can be either on or off, and the state of the next iteration of the grid is decided by a set of rules applied to the current active cells in the grid. The idea was though to use a triangular instead of a rectangular grid.

Upon further exploration the Game of Life did not translate well to non rectangular grids. It was thus chosen to concentrate more on the visual aspects of this effect. When a person touches one of the triangles it becomes activated, and then gains colour for a short duration. Meanwhile everything is continuously moving and changing colour/ shade, to create an impressive display.

In the end this effect turned out not very interactive, because the person using the installation only had the ability to activate cells, while there were also many other things changing in the installation which could not be influenced by the person. This was not all bad though because it also allowed the testing of what people find more important in such an installation: the amount of control, or visual effects.

Effect C: Begone polygon



The third effect plays with the idea that there is not a representation of your body in the visualisation, but the opposite. Your body is the void, where there is a lack of anything in the installation.

The effect consists of two things: A large collection of circles that almost move in a particle effect like fashion, and a big web of lines, which is a single large loop of line with both the ends connected. Both of these are walking around in a somewhat random fashion, moving over the screen. By touching the geometries you make them disappear.

Effect D: Oldskool

The fourth and last effect was meant to be just fun and look good without being to complicated. The person in the installation sees their body in the installation in a bright colour. A few times per second a snapshot of the people is made which gets drawn on the background. The background is stored and slowly zooms in to not only show you a direct representation of yourself, but also what you looked like a few moments ago. The colours were chosen to be random, but to all lie in the same range of brightness and feel, so as to not make certain colours stand out more than others.

Final Product

The finished product exists of two separate applications. Connected over a TCP connection. The visualisation application consists of displaying the four effects shown in the previous chapter. They rotate every minute, so people interacting with the installation for four minutes get to see all of them. If a specific effect is desired the keys 1-4 can be pressed corresponding to the four effects, to rotate to that effect immediately. Other than this there are no actual control parameters that can be adjusted in the application. All the settings can be found in the Kinect control application.

Kinect control application

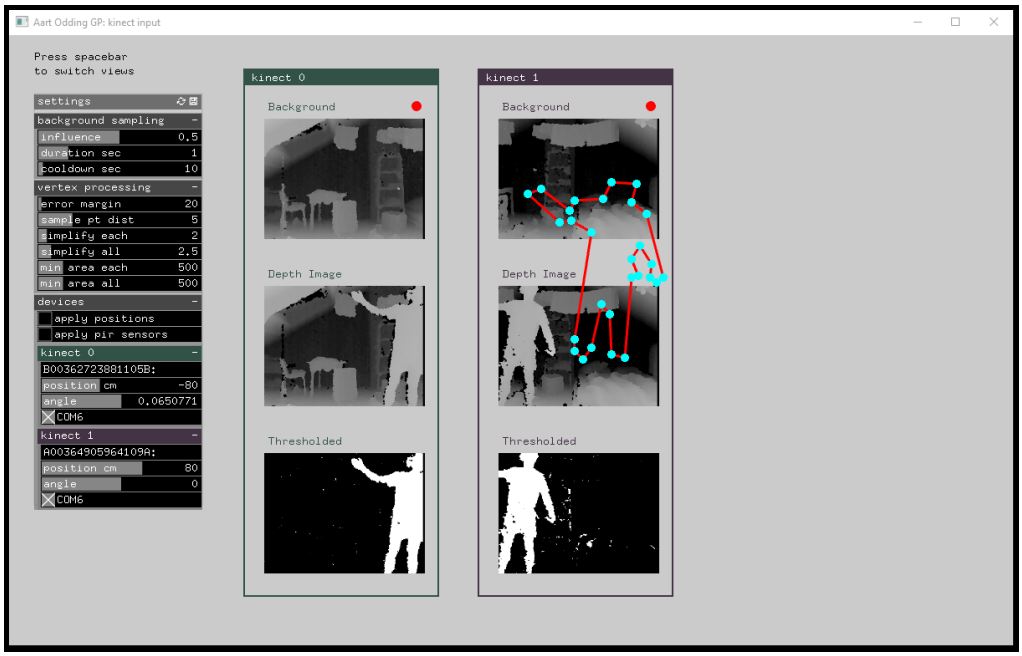


Figure 16: The finalized UI of the Kinect control application.

The Kinect control application is really the heart of the installation. The steps that are necessary to transform the pixel input that is received from the Kinects into the perpective-removed vertex data that is fed into the visualisation application as described in chapter 7 will not be discussed again, however the settings that can be controlled through the UI (figure 16), do influence this process.

In the centre of the application the input of two connected Kinects can be seen. Kinect 0 has been bordered by the dark green colour, and Kinect 1 by purple. Inside of the borders four things can be seen. The live input image coming from the Kinect. The current background image that is being subtracted from the live image. The subtracted image after a threshold has

been applied. And lastly the little red light on the top right of the area. This light tells you wether or not the current live image will also be used as background for subsequent images. Because there is a person in the image which should not be part of the background, the light is on red instead of green.

On the left side of the application are all the settings. Controlling the input processing pipeline, starting with the background samples. Updating the background of a Kinect is done using samples. Samples have a duration, influence and cooldown, each of which can individually be controlled.

The first item under vertex processing is the error margin. By lowering the error margin it is possible to get rid of the spikyness of the output outlines, however when a person has a large difference in furthest and nearest body with respect to the Kinect then their body will get distorted. Therefore it is not always best to just put the error margin as low as possible.

Sample point distance governs how far into the body, the body should be sampled for the distance to the Kinect. Make this distance too small and you risk missing the body. Make this distance too large and you risk overshooting the body in thin places like arms and hands. Next are the settings for simplification, as discussed in chapter 7, and two sliders that determine the minimum area in pixels of blobs before they are considered people.

Under devices you can see all the currently connected Kinects. For each you can adjust the angle, which controls a little motor in each Kinect, and the distance (which is because the program doesn't know how far the they are apart). Furthermore under each Kinect there is a list of all the connected PIR sensors (in this case only COM6). By enabling and disabling them it is possible to have different Kinects use different PIR sensors to sense motion. This is useful in a very large setup with many Kinects.

Lastly there is the red and cyan outline visible overlaying the Kinect images. This is the total finished result of all the processing steps, and the data that will be send to the other application.

Final setup

The finished project was setup in the SmartXP laboratory, as seen in figure 17. A large screen was used that was lowered to be at floor height, so that people can come very close to the screen. A beamer was suspended above the play area, shining down on the screen, this brought down shadows on the screen to a minimum. Two Kinects were used to sense the whole width of the area, and a single movement sensor was used to identify if there were any people.

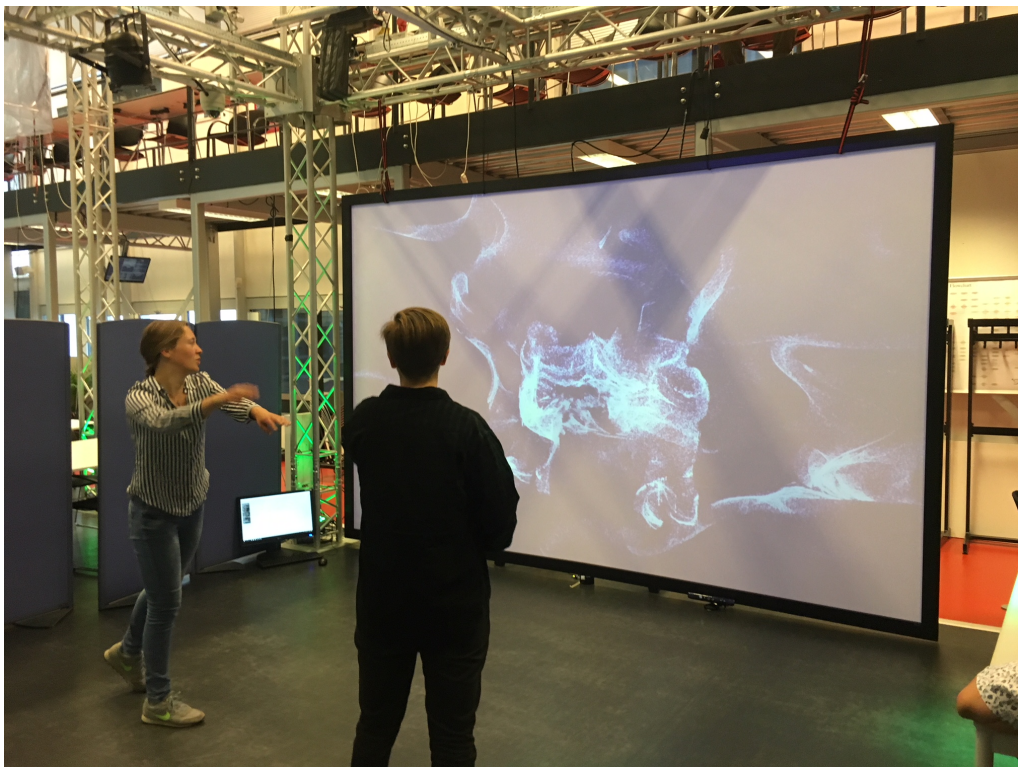


figure 17: The final setup in the SmartXP.

User Tests and Research

The main aim of the user tests, was to research what people's preferences are in this sort of interactive installation. The test was kept relatively short (one double sided sheet of a4), so that the threshold for people to participate was not too high, and people answer more serious than if the form was very large, and people want to be done quick. The questions were as follows:

1) *Please order the four effects from liked most to liked least.*

2) *What was it about your favourite effect that made you like it most?*

3) *What was it about your least favourite effect that made you like it least?*

4) *Which effect did you think was most visually pleasing?*

5) *Which effect did you think felt most interactive?*

6) *Did you think effect D was interactive? Why/ why not?*

7) *What do you think are the requirements for something to be interactive?*

8) *When playing with this installation, what would you say was more important for your enjoyment: the visuals or the interactivity? (provided with a 10 step scale from visuals to interactivity)*

9) *Space for suggestions*

It was chosen to ask the questions in such an order that the participant having just finished playing with the installation, first has to order all the effects from most to least fun. This ensures that the other more specific questions do not interfere with this intuition.

The real crux of the questionnaire is the last non open question: question 8. It is not only useful in the fact that it'll give us information on what the general public likes to see in such an interactive installation: interaction or visual effects, but also allows us to look back to question 4 and 5, where people have filled in which effect they thought was most interactive, and which one was most visually pleasing. With the combination of these two pieces of information, we can look back at question 1, to see if these preferences are also visible in their intuitive opinion of the effects. Because of this it is so important that the first question gets asked first.

Another interesting point is that according to Smuts' definition of interactivity something which is completely controlled is not interactive. With all the effects some of the behaviour comes from the computer, and some comes from people's interaction with the computer. Only effect 4 is different. Apart from the little sensor mistakes made by the Kinect the effect is completely deterministic. It'll be interesting to see how outside people think about this matter, with question 6 & 7.

The remaining open questions 2, 3 & 9, do not serve any specific purpose, and are just there to see if people have interesting thoughts on this.

In the end the test was done by 40 different participants. The full test can be found in Appendix A.

Results and Conclusion

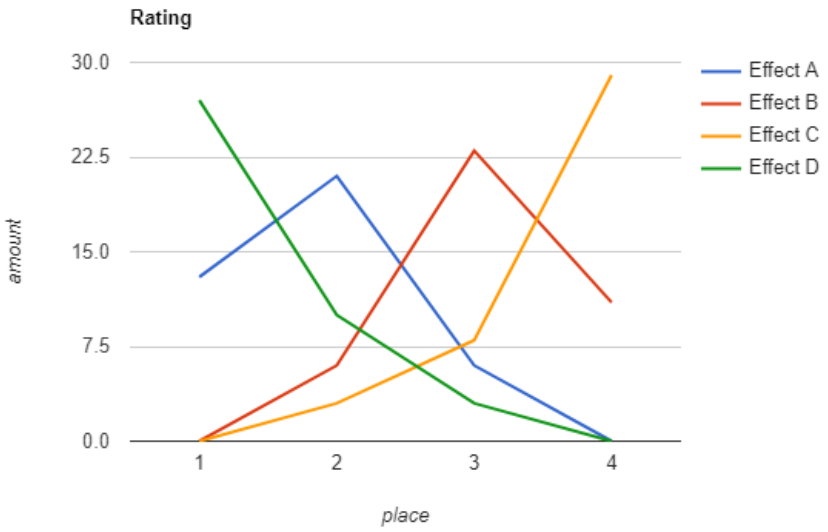


Figure 18: Ratings of each effect: 1-4 signifies 1st till 4th place.

Opinions on effect A:

+ Peaceful
+ Good that you see your own body
+ More movement creates more particles (incentive to move)
+ Elegant
- Direction of particles cannot be influenced
- Lack of colour

Opinions on effect B:

- Too few triangles (low resolution, makes it inaccurate)
- Not very interactive (only the colour changes)
- Rather abstract (not good image of yourself)
- Impact of visualisations and interactivity out of balance
- Made some people nauseous

Opinions on effect C:

- Unclear interaction (many people felt this)
- Lack of colour, visually not interesting
- Feels random instead of interactive
- No image of yourself in front of you
- Not impressive or exciting
- “Feels like a bad trip”

Opinions on effect D:

+ Actions are more lasting
+ Feels very interactive
+ Clearly see yourself in front of you
+ Lots of control
+ Looks good

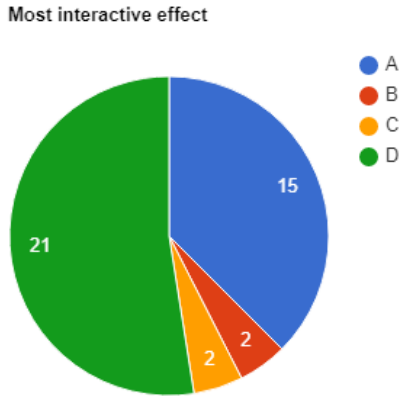


Figure 19: Amount of times each effect was chosen as most interactive.

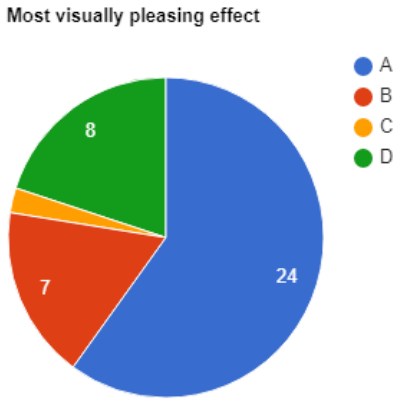


Figure 20: Amount of times each effect was chosen as most visually pleasing.

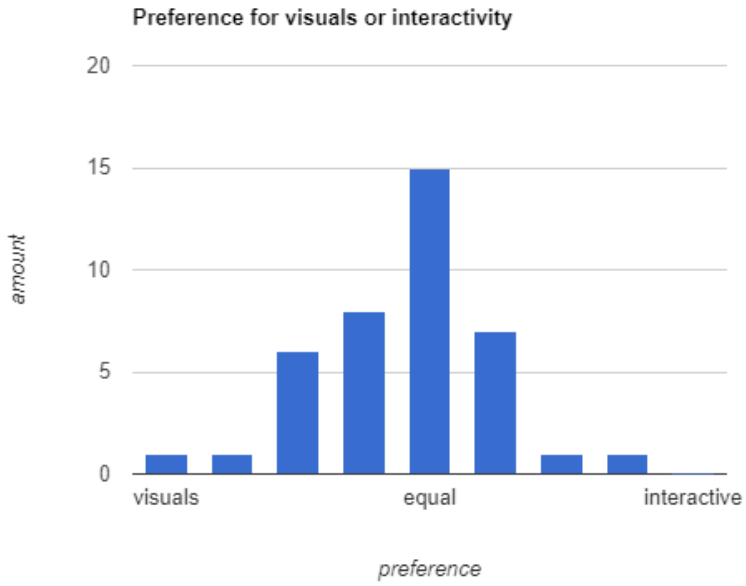


Figure 21: Participant's preference towards visuals or interaction, x = preference, y = amount of people.

1st choice	2nd choice	3rd choice	4rd choice	best visual	best interaction	V/I preference
D	A	C	B	D	A	-4
D	A	B	C	A	D	-3
A	D	B	C	A	D	-2
D	A	B	C	A	D	-2
A	D	C	B	A	A	-2
D	A	B	C	A	D	-2
A	D	C	B	A	A	-2
D	B	A	C	B	D	-2
D	A	C	B	A	D	-1
D	A	B	C	A	D	-1
A	D	B	C	A	A	-1
D	A	B	C	A	D	-1
A	D	C	B	A	A	-1
A	B	D	C	B	A	-1
D	B	A	C	D	A	-1
A	D	B	C	B	C	-1
A	D	C	B	A	A	0
D	B	A	C	D	D	0
D	A	B	C	D	D	0
D	A	B	C	A	D	0
D	A	B	C	A	D	0
D	A	B	C	D	A	0
D	A	C	B	A	D	0
D	A	B	C	D	A	0
A	D	B	C	A	A	0
D	A	B	C	A	D	0
D	A	B	C	A	D	0
D	A	B	C	B	D	0
A	C	D	B	A	C	0
D	A	B	C	B	D	0
D	A	C	B	A	A	0
D	C	A	B	D	A	1
D	B	A	C	A	D	1
A	D	B	C	A	A	1
D	A	B	C	D	B	1
D	B	A	C	B	D	1
D	A	B	C	B	D	1
D	A	B	C	A	D	1
A	D	B	C	A	B	2
A	C	D	B	C	A	3

Figure 22: Overview of all 40 participant's preferences on the effects. V/I preference stands for how much participants leaned towards preferring visuals (-4) to interactions (4).

Figure 22 contains an overview of the preferences of all 40 participants. The right most column contains their opinion on importance of visuals compared to the importance of interaction, where -4 means only visuals is important, 4 means only interactions are important, and 0 means both are equally important. The yellow markers indicate a contradiction between two or more of the answers. For instance a person with a preference for interaction (1) found B the most interactive, but in his ordering of all the effects B was third.

(Q6) Is effect D interactive?

Out of 40 people, 35 said D was interactive, one person doubted, and four people said it is not interactive.

The people that said it was interactive did not really have arguments, on why their opinion was such. But most people generally answered “yes”, or answered something in the likes of:

“Yes, because it makes you move.”

“Yes, because I can see my history”

The person that was not certain said it was down to the definition of interaction:

“It is interactive in that it responds to your movement, but I would say whether it is interactive depends on the definition of the word interactive.”

The people that deemed the effect not interactive, all had a similar reasoning as Smuts [7]:

“Not interactive, it seems to copy your shape, but not interact with any objects/ particles”

The installation copies your shape, and does something with that, however your shape is the only thing in the installation, your shape does not have any power to interact with anything.

Lastly another interesting result was that multiple people actually called D the most interactive effect. Their reasoning came mostly from the fact that the influence the participant has is so direct and visible, and from the fact that this effect made people generally very active

(Q7) What are the requirements for interactivity?

26 out of 40 people just said something along the lines of "Have some control on the visual effects.".

Six people mentioned that it is important that this response to your actions happens immediately.

Six people mentioned that the interaction between both parties should be mutual.

Five people phrased their answer such that it is dependant on the definition of interaction.

And lastly three people gave an answer that made not much sense in the context. A full list of the answers can be found in Appendix B. Some of the interesting answers included:

"Being able to change the something and the something being able to show you changes, to change your behaviour."

"A user must feel connected → action, reaction"

"Whether it is manipulatable."

"Responding in a lively way."

"See which impact your actions have on something, this can be subtle as long as people notice it."

Discussion

Measuring the finished installation against the initially devised requirements, the installation satisfies on most departments: most of the requirements have been fully achieved. The functional requirements specifically: the Kinect application supports all the features that were desired. Multiple Kinects can be connected even when they are standing at arbitrary positions. Because the installation as a whole runs as two separate applications it is very easy to reuse the Kinect part to work in conjunction with another project or installation, or even with more than one installation or program at once. All of the settings governing the processing of the Kinect data can be adjusted at runtime, which provides a lot of flexibility in its usage, and allows the user to setup the Kinect part of the installation at any location with any particular configuration with relative ease, and up to four Kinects simultaneously.

Interaction-wise most major requirements have been accomplished, however trade-offs had to be made. As described in "Realisation – Gestures" in the section about gesture recognition two different attempts have been made at creating the desired interaction. Initially it was attempted to identify specific gestures which could trigger certain effects. This idea was then discarded, because it would: A. Be very very difficult for people to realise what gestures can be recognised, and what these gestures trigger. And B. Not give people a sense of their own body being an entity inside of the installation.

Eventually the entity approach was used, where people would first and foremost see their own representation in the installation, and then use their representation to interact with the installation. However one mistake was made at this point. Most of the effects ended up following the same structure: A certain entity or group of entities is existing on the screen. At the positions where your body touches these entities, they are distorted in one way or another. The result of this is that your interaction is very location based. Using your body you can influence the installation in where a certain effect will be distorted. But in the end it is not truly possible to have any influence on the behaviour of the effect itself. The best would have likely been to have a combination of motions influencing the effect, but only have these interactions happen where a person's virtual representation resides.

Another factor that determined how interactions were handled in the final installation came from the software that was used to read the images from the Kinect. Libfreenect was used, this is a driver which loads the raw depth data from the Kinect in large arrays of distance values. Because of this it is

very involved, and out of the scope of this project to extract true gestures from this data. Alternatives would have been the official Microsoft kinect SDK, or OpenNI, however both of these have their own issues as well (License, restriction on amount of Kinects, stability issues etc.), therefore libfreenect was chosen, for its overall features and permissive license.

The final installation consisted of four different effects running in a loop, being alternated each minute. Because of this the diversity in interaction and visuals was very high, which was well received by the test subjects. The one minute timer to switch effects seemed to be a good trade-off in keeping people playing for as long as possible, and keeping people entertained by the installation. The only problem was the abrupt change of effects. Having a nicer smooth transition between the effects would have led to the installation feeling more like one experience instead of four separate effects.

One of the notable findings of the user tests was that people focus very much on the speed at which the installation responds to their interactions. People liked seeing their own reflection as clear as possible, and seeing their influences straight away. This accounts for the popularity of effect D, where the interactions where immediate, and people's mirror image was very clear. This also explains why some people considered D the most interactive effect, from all four, because of this immediacy of the control.

However it is still up for discussion whether or not effect D is truly interactive. In effect D snapshots are created using the silhouette of your body. These continuously increase in size until they are so large that it is not recognisable any more as a silhouette. People are able to fully control their virtual image in the installation, however their image is the only thing in the installation. This means that their image has nothing to interact with, and thus can't be interactive either.

This results in one of the most surprising conclusions from this research. From the start it has been assumed that the amount of interactivity in the installation should be maximized, reasoning that this would create the most fun and entertaining installation. However most people interacting with the installation have a different opinion on what is interactive, than what is considered interactive in the literature. The general public prefers to clearly see their own actions in the installation, and this is where their sense of interaction comes from. Most people would be happy with an installation that has great visuals, and a certain degree of control the user can exert. So what then should be strived for when creating an interactive art installation? Just a good experience? We believe that to create the perfect interactive art installation, that is both the most fun for the user, and truly interactive needs the following aspects:

Emergent generative and complex behaviour. If you want to interact with something it needs a certain degree of complexity to be able to meaningfully influence with it. Just like how as a human you could have more meaningful and genuine interactions with a cat or dog than with a spider.

Long term influence and short term influence. Short term to feel interactive instantly, and long term to feel like your interactions are meaningful: similarly to an interaction between humans has long and short term effects (short term: The successive responses on each other, long term: Changing each other's opinion on a certain topics) interactive art also needs interactions on both time scales.

The right balance of control. This is not as straight forward as it sounds: Smuts argues: an interaction should not be completely controlled or completely controlling, however during the user tests it has appeared that most people prefer to have as much control as possible in the installation. This is therefore a topic worthy of future research. One good approach would be to take over control during an interaction, while when not (actively interacting) the installation tries to take control back from the user. Again this should be attempted and be experimented with.

Change of interaction, or change of influence of interactions. If one can interact with an installation by lifting one’s arm, and each time the installation responds with the same outcome the installation will quickly become boring. This comes back to the same point as the first requirement: sufficiently complex behaviour. Which would cause interactions to have different outcomes each time. Think of a person A saying the same sentence every day to a person B. Depending B’s mood, A and B’s relationship and infinitely more factors, the response would be different every day.

Future Research

Interactive art is a still a new and emerging field, with many aspects left to explore. This thesis is hopefully helpful for those who wish to create their own interactive art, or wish to research interactive art themselves.

The usability test of this project was mostly done by university students aged 18 – 30. The wants and expectations of this group might not accurately represent the wants and expectations of the general public. More research could be done to the perception of interactive art by different social groups, interacting alone or in groups, with someone to explain what the installation is or completely unguided, etc.

Of course this project was one very specific installation, and therefore might not accurately represent all interactive art in general. A larger sample size and more interactive art installations are always desired.

Furthermore the complete dimension of audio has not been touched in this project, which would have a large impact on how people interact with and perceive the installation.

Interactive art is still young, and most interesting research and projects are still to come.

References

[1] M. Duchamp, *Rotary Glass Plates (Precision Optics)*. New Haven, Connecticut: Yale University Art Gallery, 1920.

[2] Y. Agam, *Transformable Relief*. Private collection, 1953.

[3] B. Fry and C. Reas, *Processing*. Processing Foundation, 2001.

[4] M. Banzi, D. Cuartielles, T. Igoe, G. Martino and D. Mellis, *Arduino*. Arduino LLC, 2018.

[5] D. Lopes, "The Ontology of Interactive Art", *Journal of Aesthetic Education*, vol. 35, no. 4, p. 65, 2001.

[6] S. Cornock and E. Edmonds, "The Creative Process Where the Artist Is Amplified or Superseded by the Computer", *Leonardo*, vol. 6, no. 1, p. 11, 1973.

[7] A. Smuts, "What Is Interactivity?", *The Journal of Aesthetic Education*, vol. 43, no. 4, pp. 53-73, 2009.

[8] TeamLab, *Wander through the Crystal Universe*. 2016.

[9] T. Lengeling and G. Castro, *Aether*. 2013.

[10] C. Bardainne and A. Mondot, *Hakanaï*. 2013.

[11] TeamLab, *Drawing on the Water Surface Created by the Dance of Koi and People - Infinity*. 2016.

[12] Potion Design, *Forest Friends*. 2015.

[13] N. Selikoff, *Beautiful Chaos*. 2013.

[14] H. Martin, *libfreenect*. OpenKinect Project, 2010.

[15] *OpenCV Library*. Intel Corporation, Willow Garage, Itseez, 2000.

[16] B. Gehrels, B. Lalande, M. Loskot and A. Wulkiewicz, *Boost.Geometry*. Boost Organization, 2009.

[17] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves", *Computer Graphics and Image Processing*, vol. 1, no. 3, pp. 244-256, 1972.

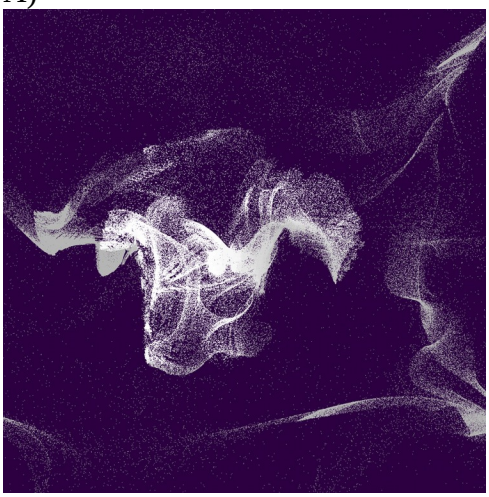
[18] D. Douglas and T. Peucker, "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature", *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112-122, 1973.

[19] H. Blum, "A transformation for extracting new descriptors of shape", *Models for the Perception of Speech and Visual Forms*, pp. 362-380, 1967.

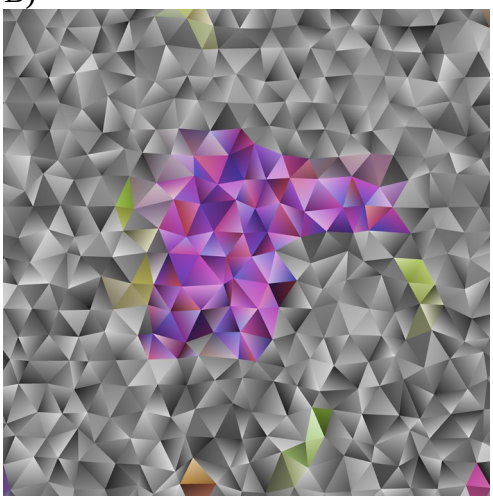
[20] O. Aichholzer, F. Aurenhammer, D. Alberts and B. Gärtner, "A Novel Type of Skeleton for Polygons", *J.UCS The Journal of Universal Computer Science*, pp. 752-761, 1996.

Appendix A – User Test

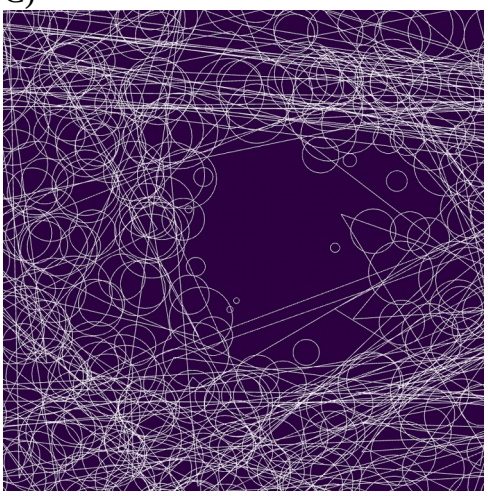
A)



B)



C)



D)

How old are you? . Gender .

Please give the order of your preference to which effects you liked most, to which you liked least.

Liked most			liked least

What was it about your favourite effect that made you like it best?

What was it about your least favourite effect that made you like it least?

Which effect did you think was most visually pleasing?

Which effect did you think felt most interactive?

Did you think effect (D) was interactive? Why/ why not?

What are your requirements for something to be interactive?

When playing with this installation, what would you say was most important for your enjoyment: the visuals, the interactivity, or both equally.

Visuals				equal				Interactivity

Any particular reason why?

Would you say that any of the interactions could have been stronger/ done in a different way?

Do you have any ideas/ tips to improve this installation?

Appendix B – Answers to Open Questions

(Q6) Is effect D interactive?

"Yes, snapshots of what you are doing, feels like a stroboscope."

"Yes, I played with the installation the most with this effect."

"I did, I could directly see my movements on the screen."

"Yes, you could take on poses and see yourself back on the screen very clearly."

"Kind off, it was more like snapshots where taken, instead of really changing the visualisation."

"Yes, it looks so cool, it moves with you."

"Yes, I think it was the most interactive as I could see an effect."

"Yes, because you can see the movement and history."

"Yes, I see a silhouette of my movements/ position."

"Yes, you could actually see a silhouette."

"Yes!"

"Yes, seeing your movement update is awesome."

"Not interactive, it seems to copy the shape, but not interact with any object/ particles."

"Not really because it is just a live visualisation."

"A little, because it changed on my movement but it didn't do much more."

"Not really since the movement was tracked by all the effects. D had no other additions."

"It is interactive in the way that it responds to your movement, but I would say it depends on your interpretation of the word interactive."

"Yes, effect changes on movement."

"For some reason it really encourages the user to interact with it. It has a 'just dance' kind of feel."

“Yes, you made the screen show up with a different picture depending on your pose, coming up with something different every time.”

“A bit slow reaction, but that made it nice.”

“Yes, because you could clearly see it was you.”

“Yes, however it seemed like short snapshots after each other.”

“Yes, it copied my movements, and distorted my physique.”

“Yes, it used your body shape to show cool visuals.”

“Yes, the program still mirrored my movement. It was even more interactive than B.”

“Yes, since the shapes change and you have influence on the at.”

“Yes, you saw yourself coming back in the shapes.”

“Yes, because it showed your movements.”

“It imitated: very direct interaction.”

“Yes, it changed very well with the movement of the user and adapted to the poses well.”

“Yes, your acting shapes what happens on the screen.”

“Yes, I could clearly see myself and it was quite cool.”

“Yes, your posture is really clearly visible.”

“Yes, it was clear what pose you made.”

“Yes, it is very clear how your interactions translate to the shapes on the screen.”

“Yes, because you can also see your history (and order of movements/ the different layers).”

“Yes, it copied movement and dared you to try to create new shapes.”

“Yes, because the effect moves to the outside borders, you go there to try things out.”

“Yes, because it was most realistically shaped and made it easy to interact with others.”

(Q7) What are the requirements of interactivity?

“I move it updates.”

“Status of the machine changes accordingly to user input.”

“Fast reaction.”

“I can clearly see the changes that I cause.”

“See which impact your actions have on something, this can be subtle as long as people notice it.”

“I can so something with the presented technology.”

“It reacts to what you’re doing (your input).”

“Being able to change the something and the something being able to show you changes, to change your behaviour.”

“There has to be a reaction to an action you perform.”

“I should see change when I do something.”

“That I am interested.”

“Clear feedback of your actions.”

“To give a reasonably accurate impression of a human being.”

“Two units that respond to each other continuously.”

“Engagement to try and create new objects/ things.”

“Control on the effect by the user, and vice versa.”

“You should be able to understand what happens on the screen.”

“My actions affect the something.”

“It has to be inviting, and putting in more effort should result in more awesome stuff.”

“That I am given some control over the installation.”

“My actions have impact on the installation.”

“Something has to change as soon as the user makes an effort to interact with the installation.”

“Direct and natural interaction generates action.”

“That it reacts on what you do, that you see it follows you E.G.”

“Seeing your interaction fast.”

“You have influence and can change what is happening.”

“You can do something.”

“Responding to your actions.”

“If I approach, touch or do something, the installation should react to me.”

“Interaction with user.”

“To react on what you are doing.”

“Quick responses.”

“Whether it is manipulatable.”

“Visually appealing, and the effect has to be noticeable straight away.”

“A user must feel connected → action, reaction.”

“Responding to your input.”

“Aspects of a program that respond in a lively way. Meaning that I can see that it does more than just track my movement.”

“That it does something more than just copy my movement, but does something else with it.”

“In this case something should react to your movement: like in case C where circles get smaller and disappear.”

“Depends on the definition of interactive, I guess you have to be able to move or change objects.”

Appendix C – Shaders

Effect A – Vertex Shader

```
#version 330 core

layout (location = 0) in vec2 position_in;
layout (location = 1) in float ttl_in;
layout (location = 2) in float time_offset;

out vec2 position_out;
out float ttl_out;

uniform float time;
uniform float elapsed_time;
uniform float speed;

uniform vec2 window_size;
uniform vec3 noise_scale_inside;
uniform vec3 noise_scale_outside;

uniform sampler2D stencil;


void main()
{
  vec3 noise_pos;
  vec2 sample_pos = vec2(
    position_in.x / window_size.x,
    (window_size.y - position_in.y) / window_size.y );

  if (texture(stencil, sample_pos).x > 0)
  {
    ttl_out = 0;
    noise_pos = vec3(position_in, 1.0f) * vec3(
      noise_scale_inside.x, noise_scale_inside.y,
      (time + time_offset) * noise_scale_inside.z);
  }
  else if(ttl_in < 1.0f)
  {
    ttl_out = ttl_in + elapsed_time;
    noise_pos = vec3(position_in, 1.0f) * vec3(
      noise_scale_inside.x, noise_scale_inside.y,
      (time + time_offset) * noise_scale_inside.z);
  }
  else
  {
    ttl_out = ttl_in + elapsed_time;
    noise_pos = vec3(position_in, 1.0f) * vec3(
      noise_scale_outside.x, noise_scale_outside.y,
      (time + time_offset * 3) * noise_scale_outside.z);
  }

  float noise_val = snoise(vec3(noise_pos.x, noise_pos.y, noise_pos.z)) * 4.5f;
  vec2 movement = vec2(cos(noise_val), sin(noise_val)) * speed;
  position_out = position_in + movement;

  if (ttl_in < 1.0f || time_offset > 4)
  {
    gl_Position = vec4(map(position_out, vec2(0, 0), window_size,
      vec2(-1, -1), vec2(1, 1)), 0, 1);
  }
  else
  {
    gl_Position = vec4(-2, -2, 0, 1);
  }

  if (position_in.x < 0 || position_in.x > window_size.x ||
      position_in.y < 0 || position_in.y > window_size.y)
  {
    position_out.x = random(vec2(position_in.x, time)) * window_size.x;
    position_out.y = random(vec2(position_in.y, time)) * window_size.y;
  }
}
```

Effect A – Fragment Shader

```
#version 330 core

out vec4 frag_color;

in float ttl_out;


void main()
{
  if (ttl_out < 1)
  {
    frag_color = mix(vec4(1, 1, 1, 1), vec4(0.7, 0.7, 0.7, 1), ttl_out);
  }
  else
  {
    frag_color = vec4(0.7, 0.7, 0.7, 1);
  }
}
```

Effect B – Vertex Shader

```
#version 330 core

layout(location = 0) in vec2 position_in;
layout(location = 1) in vec2 center;
layout(location = 2) in vec2 time_offset;
layout(location = 3) in float random_walker;

layout(location = 4) in float hue_a;
layout(location = 5) in float hue_b;
layout(location = 6) in float saturation;

out float hue_a_out;          // captured
out float hue_b_out;          // captured
out float saturation_out;     // captured

out vec3 vertex_color;

uniform vec2 window_size, current_hues, pt_ab, pt_cd;
uniform float current_time;
uniform float delta_time;
uniform sampler2D stencil;


void main()
{
  bool s1 = texture(stencil, vec2((center.x + pt_ab.x) / window_size.x,
    (window_size.y – (center.y + pt_ab.y)) / window_size.y)).x > 0;
  bool s2 = texture(stencil, vec2((center.x - pt_ab.x) / window_size.x,
    (window_size.y - (center.y - pt_ab.y)) / window_size.y)).x > 0;
  bool s3 = texture(stencil, vec2((center.x + pt_cd.x) / window_size.x,
    (window_size.y - (center.y + pt_cd.y)) / window_size.y)).x > 0;
  bool s4 = texture(stencil, vec2((center.x - pt_cd.x) / window_size.x,
    (window_size.y – (center.y - pt_cd.y)) / window_size.y)).x > 0;

  float gray_noise = (snoise(vec3(position_in, current_time + time_offset))
    + 1) / 2;
  float hue_noise =  (snoise(vec3(position_in, current_time + time_offset
    + 1000)) + 1) / 2;

  if (s1 || s2 || s3 || s4)
  {
    hue_a_out = current_hues.x;
    hue_b_out = current_hues.y;
    saturation_out = 0.5;
  }
  else
  {
    hue_a_out = hue_a;
    hue_b_out = hue_b;
    saturation_out = max(saturation - (delta_time / 2), 0);
  }

  float hue = abs(fract(hue_a + hue_noise / 2));

  vertex_color = hsl2rgb(vec3(hue, saturation, gray_noise));
```

```
if (random_walker > 0 && saturation == 0)
  {
    hue = abs(fract(snoise(vec3(center.x / 1000, center.y / 1000, current_time
      / 100)) / 2));
    vertex_color = hsl2rgb(vec3(hue, random_walker, gray_noise));
  }

  float dir = snoise(vec3(position_in.x / 100, position_in.y / 100,
    current_time / 10)) * 4.5;

  float mag = snoise(vec3(position_in.x / 100, position_in.y / 100, 100 +
    current_time / 10)) * 30;

  gl_Position = vec4(map(position_in + vec2(cos(dir) * mag, sin(dir) * mag),
    vec2(0, 0), window_size, vec2(-1, -1), vec2(1, 1)), 0, 1);
}
```

Effect B – Fragment Shader

```
#version 330 core

in vec3 vertex_color;

out vec4 out_color;


void main()
{
  out_color = vec4(vertex_color, 1.0);
}
```

Effect C – Vertex Shader for Circles

```
#version 330 core

layout (location = 0) in vec2 coord_in;
layout (location = 1) in vec2 centre_in;
layout (location = 2) in float shrinking_in;
layout (location = 3) in float noise_offset;

out vec2 coord_out;
out vec2 centre_out;
out float shrinking_out;


uniform vec2 window_size;
uniform float time;
uniform sampler2D people;



void main()
{
  float x = snoise(vec3(1, 1, time * 0.1 + noise_offset));
  float y = snoise(vec3(1, 1, time * 0.1 + noise_offset + 300));

  vec2 movement = vec2(x, y) * 2.5;

  coord_out = coord_in;
  centre_out = centre_in + movement;

  if(centre_out.x > window_size.x + coord_out.y) {
    centre_out.x = -coord_out.y;
  } else if(centre_out.x < -coord_out.y) {
    centre_out.x = window_size.x + coord_out.y;
  }
  if(centre_out.y > window_size.y + coord_out.y)
  {
    centre_out.y = -coord_out.y;
  } else if(centre_out.y < -coord_out.y) {
    centre_out.y = window_size.y + coord_out.y;
  }

  float two_pi = 6.28318530717958647693;

  bool hit = texture(people, vec2(centre_out.x / window_size.x,
    (window_size.y - centre_out.y) / window_size.y)).x > 0;

  for(float a = 0; a < 6.25; a += two_pi / 8)
  {
```

```
    vec2 pos = centre_out + vec2(sin(a) * coord_out.y * 0.5, cos(a) *
      coord_out.y * 0.5);

    if (texture(people, vec2(pos.x / window_size.x, (window_size.y - pos.y)
        / window_size.y)).x > 0)
    {
      hit = true;
    }
  }

  if (hit)
  {
    coord_out.y -= 5;

    if (coord_out.y < 0)
    {
      centre_out.x = random(centre_in.x);
      centre_out.y = random(centre_in.y);
      coord_out.y  = 40 + 40 * random(centre_in);
    }
  }

  vec2 position = vec2(cos(coord_in.x) * coord_in.y, sin(coord_in.x) *
    coord_in.y) + centre_out;

  gl_Position = vec4(map(position, vec2(0,0), window_size, vec2(-1, -1),
    vec2(1, 1)), 0, 1);
}
```

Effect C – Fragment Shader for Circles

```
#version 330 core

out vec4 frag_color;


void main()
{
  frag_color = vec4(1, 1, 1, 1);
}
```

Effect C – First Vertex Shader for Lines

```
#version 330 core

layout (location = 0) in vec2 position_a;
layout (location = 1) in vec2 position_b;
layout (location = 2) in vec2 position_c;
layout (location = 3) in vec2 noise_offset;

out vec2 position_out;

uniform float time;
uniform float buffer_size;
uniform vec2 window_size;
uniform vec3 noise_scale;
uniform sampler2D people;

bool collision()
{
  bool result = false;

  float n_times = 10;
  float inc = 1.0f / n_times;

  for (float a = 0; a <= 1; a += inc)
  {
    vec2 pos1 = mix(position_a, position_out, a);
    vec2 pos2 = mix(position_c, position_out, a);

    if (texture(people, vec2(pos1.x / window_size.x, (window_size.y - pos1.y) /
        window_size.y)).x > 0 || texture(people, vec2(pos2.x / window_size.x
       (window_size.y – pos2.y) / window_size.y)).x > 0)
    {
      result = true;
      break;
    }
  }
  return result;
}
```

```
void main()
{
  float x = snoise(vec3(1, 1, time * noise_scale.z * 0.6 + noise_offset.x
    + 100));
  float y = snoise(vec3(1, 1, time * noise_scale.z * 0.2 + noise_offset.y
    + 300));

  vec2 mov = vec2(x, y) / length(vec2(x, y)) * 4;

  position_out = position_b + mov;

  if (position_out.x < -buffer_size || position_out.x > window_size.x +
      buffer_size)
  {
    position_out = vec2(random(position_out) * window_size.x,
      random(position_out.yx) * window_size.y);
  }

  if (position_out.y < -buffer_size)
  {
    position_out = vec2(position_out.x, window_size.y + buffer_size);
  }
  else if (position_out.y > window_size.y + buffer_size)
  {
    position_out = vec2(position_out.x, -buffer_size);
  }

  int n = 0;

  while (collision() && n++ < 7)
  {
    position_out = vec2(random(position_out) * window_size.x,
      random(position_out.yx) * window_size.y);
  }

  gl_Position = vec4(-3, -3, 0, 1);
}
```

Effect C – Second Vertex Shader for Lines

```
#version 330 core

layout(location = 0) in vec2 position_in;

uniform vec2 window_size;


void main()
{
  gl_Position = vec4(map(position_in, vec2(0, 0), window_size, vec2(-1, -1),
    vec2(1, 1)), 0, 1);
}
```

Effect C – Fragment Shader for Lines

```
#version 330 core

out vec4 out_color;

void main()
{
  out_color = vec4(1.0, 1.0, 1.0, 1.0);
}
```

Effect D

For effect D only the default rendering pipeline present in OpenFrameworks was used without the addition of custom shaders.