Semester 3 Report

Group 307:

Boris Lykke Nielsen, Casper Sørensen, Stefan Lavlund Skau, Emil Rose Høeg
Patrick Øllegaard Chieu, Martin Nygaard, Frederik Ditlev Frøling

November 9, 2013

Contents

1 Introduction

The video game industry has evolved a lot over the past few decades. Video games are constantly getting larger in size, gaining more functionality, and the graphical fidelity reaches new heights with every new release. This evolution is possible because the hardware on which the games are run is also evolving and getting more and more powerful.

One aspect of the video gaming industry which has not changes considerably since the beginning is the controllers used by the different platforms. All the leading video game consoles use a variation of the same basic design: a hand-held controller with one or more analog sticks and a set of buttons for the user to interact with. Sonys PlayStation has used the same design of the controller since the first PlayStation was introduced. The same goes for Microsofts Xbox. The personal computer (PC) has used the basic keyboard and mouse for controls for many years now and it remains the preferred control method.

This has changed over the past years however, as the living room consoles have developed new ways of interacting with video games. These new methods include the usage of visual computing and one or more cameras to read the user, which means that the user now uses body movements instead on thumb presses to control the games. This introduces new ways of video game interaction and paves the way for a new breed of video games.

One platform which is yet to see any development in its control scheme is the PC which still rely on the same digital keyboard and mouse controls. The digital input from a keyboard does not translate well into vehicle games like racing games where the vehicles do not go all the way left or all the way right. This means that vehicle games are often best experienced on the living room consoles where the analog sticks on the controllers can be better used to fine tune the movement of the car. This has led to the development of add-on steering wheels and analog joysticks which can be bought together with vehicle games and plugged into the PC. However these controls often add cost to the PC game and are generally not easy to manage on the desk in front of the computer once you are done playing the game.

Many PCs these days are equipped with webcams in the screen for use in video conversations so the idea of using these webcams together with visual computing to get input from the user could be transferable from the living room consoles to the PC as an alternate method of controlling these vehicle based games for an experience comparable to those provided by the living room consoles.
This leads to the following initial problem statement.

1.1 Initial problem statement

How can a vehicle-game controller be made from a standard webcam using visual computing, comparable with other gaming platforms utilizing movement based functionalities?

2 Methods

2.1 Naïve designs

The naïve designs are, as the name implies, designs based on assumptions and ideas, instead of actual research or considerations concerning feasibility taken into account. The designs are not meant to be used as a final design, but rather as inspiration source for which areas needs further research, before finalizing the design.

The designs start as a brain storm, revolving around ways in which the initial problem statement could be answered. The ideas from this brain storm will then be used to create some design concepts, explaining how a product can be created, how it is intended to work and how it is used to answer the problem. Once the designs are made, thought will go into which elements needs further research, in order to actually make the product described, and to make sure that it actually tests, what it is supposed to.

This approach of "Initial Problem Statement -> Naïve Design -> Analysis -> Final Problem Statement" replaces the "traditional "Initial Problem Statement -> Investigation -> Final Problem Statement -> Analysis", and the reason this is done, is an attempt at avoiding the risk of getting tunnel vision of a specific way to solve a problem - Instead of focusing on a single way of solving a problem right from the get go, this approach revolves around having multiple solutions in mind, right up until the point, where the best (potentially, at least) solution have been found, through the process of delimitation.

2.2 Analysis

As our group learned which subjects that should be focused on from the Naïve designs. The analysis is a simple; yet time consuming process to execute. The group got to research each focus point and get the information required in order to create a final problem statement (FPS) that will for fill our requirements to this project. The focus points in this analysis will be:

- What have others done before us?
  The group is going to research State of the Art (SOTA) to experience what others have created, to learn from their mistakes and to find inspiration for our projects.

- How does Image Processing work?
  The group is going to research Image Processing to be able to understand how the process work and be able to use the information that has been learned to create a product that works and is capable to produce data that can be used for evaluating.

- Can we form our gestures so the volunteers feel comfortable during tests?
  If the group wants results that can be used later on a good test has to be performed. In order to achieve this we want our volunteers to feel comfortable and use gestures that feel natural to them.

When these steps have been followed the group should be ready to form our FPS. The group is then going to use this FPS to create new designs based on the new information gained in the analysis and then be able to move forward to our design phase.

2.3 Design

The approach in the design chapter is to conceptualize a design based on the List of Requirements assembled through the analysis. This List of Requirements is used in the design in order

to ensure that the final product will fulfill all the demands established during the research in the analysis. During the design-process, there will be some low-fi prototype tests in order to ensure that the concept behind the design is understandable and usable by users unaware of the prior research leading to the described design.

Instead of locking onto one design, the chapter will try to conceptualize multiple design ideas, thus trying to take focus in different areas researched in the analysis. This should give the possibility to merge strong parts of either design in order to create a final design that will be implemented into the final product.

2.4   Implementation

In order to create a prototype that can be used for testing with satisfactory results, as well as serve as a baseline for future development, the final design should be implemented as a horizontal prototype. Horizontal prototyping as opposed to vertical prototyping is based on the idea of function before aesthetics (Rogers, Sharp, and Preece 2002). Horizontal prototyping means creating a prototype that has a wide range of functionality but in a very crude state. Vertical prototyping provides a lot of detail but only on a few set of features. This means that in relation to this design a software prototype will be created that will contain, close to, the full range of functionality from the final design but in a crude state that will not be user friendly or meet the performance requirements.

The implementation will consist of a single piece of software that will map the input provided by a webcam to a set of keyboard and mouse commands. This software will work in the background and provide input to any application intended for testing.

2.5   Evaluation

It is essential that the methodology of the evaluation for the product-test is defined. To achieve this, the desired outcome of the test must be clearly identified i.e. a proper strategy for evaluating whether or not, and to what degree, the final problem statement has been answered. Furthermore, the relevance of the established list of requirements must be tested.

The main criterion of the product test can be defined thus in bullet points:

- To investigate whether or not the final problem statement has been answered.

- To prove/disprove the relevance of the list of requirements.

2.5.1   Triangulation

(TO BE CONCLUDED)

> *Triangulation involves using combinations of techniques in concert to get different perspectives or to examine data in different ways.* (Rogers, Sharp, and Preece 2002)

2.5.2   Usability Test

> *"Ideally, user-based testing would take place during all stages of development, but that is not always possible. Why do we do usability testing? As much as designers try to build interfaces that match the needs of the users, the designers are not users and even the users themselves sometimes cannot clearly identify their interface needs."* (Lazar 2010)

"Many people say that five users is the magic number and that five users will find approximately 80% of usability problems in an interface (Virzi, 1992). This has become a generally accepted number in HCI, but many other researchers disagree with the assertion. The major challenge in determining the right number of users is that you don't know in advance how many interface flaws exist, so any estimate of how many users are needed to find a certain percentage of interface flaws is based on the assumption that you know how many flaws exist, which you probably don't."
(Lazar 2010)

2.5.3 Preliminary Test

(TO BE CONCLUDED)

3 Naïve designs

The way this report is approached, is slightly different than usual - Normally the problem is approached by coming up with an idea of what to work with, then do some initial research (investigation) with the goal of turning the idea into an initial problem statement, which is then used to narrow the field of research into an analysis, with the goal of turning the initial problem into a final problem statement.

This report structure is somewhat similar in content, but different in approach: Instead of starting by coming up with a general idea, the first step was to come up with an initial problem statement.

This problem statement was then used as a framework for creating some basic (naïve) designs, in order to figure out, in which direction the report should go, before doing any specific research.

These designs are then used to figure out which areas research should be focused on (i.e. they effectively work as an investigation chapter).

The reason this approach is used, is to increase efficiency and broadening the perspective of possible solutions to a problem - instead of doing a lot of preliminary research on a subject, in order to create a problem statement, the order is reversed, which reduces the amount of work that "goes to waste" (i.e. ends up not being important for the report), and instead of locking upon a single solution, this approach tries to focus on many possibilities of solving a problem.

3.1 Naïve design summaries

The naïve design ideas are presented in a short and precise manner for summary purposes. Only essential information and documentations are described. For a detailed description and explanation for each specific naïve design read appendix A.

3.1.1 Control a car through gestures - Appendix A.1

The idea is to make a vehicle controller, out of a camera, that can react to a user' hand gestures and translate that into input for a vehicle based application or game. These gestures would translate to tasks commonly used in relation to vehicle games, like speed up, slow down, change gears, hand breaking. The system should also be used to control the vehicles directions.

3.1.2 Controller Blocks - Appendix A.2

The idea is to create a vehicle controller that utilizes blocks which the players move around on a flat surface in order to navigate, along with additional functionalities, the specified vehicle game. This idea supports both single and multiplayer, for the basic standard webcam will be pointed down on the flat surface and will be able to track two players navigating their blocks separately.

3.1.3 Immersive Racing Game Experience - Appendix A.3

The idea is to create a controller that uses a standard basic webcam to track the whole body of a person, allowing movement, gestures and face recognition to control the presented game through defined gestures and movements.

3.1.4 Pointing Controllers - Appendix A.4

The idea behind this concept is to use a webcam to track finger(s) on a plain surface to register movement and functionalities of the designated game. By relocating the finger(s), depending on the game, the input will act accordingly to the specific game/functionality.

3.1.5 Object Controlled Visual Processing - Appendix A.5

The idea revolves around a multicolored - or shape-recognizable object that can be moved around in front of the webcam to create a motion capture-device that can be used as a logical vehicle controller e.g. a plate as seen in the image for controlling a car or boat etc.

3.1.6 Race-Making game - Appendix A.6

The idea is to have a controller utilizing a board where you place down the bricks as "points" where you would like the track to go. The game then connects these points with lines, and there you have a track to maneuver the vehicle inside the specified vehicle game. To control the vehicle, a user's finger is traced through a static webcam.

3.1.7 Tabletize your computer - Appendix A.7

The idea for this design is, using a standard webcam, simulating the rotation measuring devices of modern tablets and smart phones, allowing computer users the same intuitive means of controlling games.

As the webcam is recording, edges will be recognized (using image processing), and the movement of said edges will be converted into movement on the computer - In other words, movement within the captured pictures will be replacing the tilting sensors of the tablets and smart phones.

3.2 Focusing the research areas

Looking through the naïve designs (see appendix), it was noticed that most of the ideas had common research areas. So we took the best of the ideas and had a look at the requirements for each naïve idea. A list of required research areas became apparent from the studies of the naïve designs.

The list has been prioritized according to importance for the project - determined by a combination of how many of the designs listing a specific requirement and by how a general a requirement is (i.e. can it be used for a design where it is not listed initially).

The reason for prioritizing the list in such a way, is to keep it short - filling the report with research, not being used is inefficient in terms of time used and can be disruptive for the reader - because of this, "unimportant" research have been excluded (or in some cases, put into the appendix). In the appendix, a table of how each topic have been evaluated and prioritized.

The order of importance of the research areas is listed here and described below:

- SOTA (State Of The Art)

- Image Processing and Analysis (IP & IA)

- Transferring data from one application to another

- Gesture recognition

- Controller appeal / motivation of the users

- Networking

- Webcam differences

3.2.1 SOTA (State Of The Art)

To compare a new game controller with something already in production, the first step is to do research on the existing solutions. More specifically, research should revolve around how they work, why these products are popular, and try to find out how it was designed and developed, to avoid repeating their mistakes. This research can be found in the analysis (chapter: 4).

3.2.2 Image Processing and Analysis (IP & IA)

As this project is going to be heavily based on video / image processing and analysis, it is very important to research how to use the different techniques and methods associated with working in these fields. As can be seen in appendix A, most of the naïve designs focus on using color recognition, pattern recognition and edge detection - making these techniques especially important to research. Additionally, as some of the designs include the possibility to allow multiple players to play at the same time, blob detection is also important to research, as this method allows the program to distinguish between different elements in an image.

3.2.3 Transferring data from one application to another

As the focus of this project is not creating a game, but rather creating a controller for a game, a key component of designing the controller, is enabling it to transfer data from the image-capture into a 3rd party application (i.e. a game). Whether the control will transfer direct data (i.e. creates a new input for the game) or simply "mimics" existing game controls (e.g. a specific movement is turned into a specific button push on a keyboard).

To do this, some research of how to make two applications communicate accurately and efficiently is required.

3.2.4 Transferring data from one application to another

As gesture recognition is used for some of the naïve designs, this field should also be researched. While this specific method of image processing is not as important as the other, as many of the features of gesture recognition could possibly be done, using pattern recognition instead, it is still a relevant topic for this report, and as such, should be researched.

3.2.5 Controller appeal / motivation of the users

In order to improve the general user experience, some research into controller appeal and how to motivate a user should be made including - but not limited to - physical design, user interface and user controls (which gestures are most appropriate to use etc.).

3.2.6 Webcam differences

The idea with this project, as mentioned, is to make a solution that is cheap for the users, which means that, as most people already has a webcam, a webcam would not be required for the user to buy to use most of the naïve design ideas. As people might have very different webcams some webcams might require a different setup in order to be able to use the product. To be able to create a product that can be used by all kinds of webcams it is therefore required to know what distinguishes the different webcams from one another, such as how good they are at grabbing the true colors etc.

4 Analysis

The purpose of this analysis is to collect the relevant research and data which was deemed necessary during the naïve design phase. This research is going to be used as a basis for a prototype, which should help us answer our problem statement.

4.1 Product Comparisons

As stated by our initial problem, can an ordinary webcam compare to other gaming platforms. It is necessary to figure out which other gaming platforms to compare the webcam with. Since the webcam is a video camera it can be easily compared with other video game platforms that utilize a video camera. The Xbox Kinect from Microsoft utilizes one RGB-camera and two 3D depth sensors to detect gestures made by the user (Cong and Winters n.d.). The Kinect is popular, having sold roughly 24 million sensors during its current lifespan (Microsoft 2013), and because it has become such a household name, makes it a suitable platform for comparison.

Another popular brand is the Wii by Nintendo. The Wii was introduced in 2006 and was the first console to have motion control. During its lifespan the Wii has sold near 100 million units worldwide (Ltd 2013). The Wii also uses a camera but not in the same way as the Microsoft Kinect, instead of using a static VGA camera like the Kinect, the Wii uses an infrared camera positioned within the hand-held controller held by the user to detect the position of the controller in relation to the monitor (Castaneda 2006). This makes the controller usable as a pointing device. The Wii controller also uses motion sensors to detect the tilting of the controller, these allow the controller to be used as a virtual golf club or tennis racket. The Wii' use of camera technology makes it suitable for comparison.

The PlayStation Move by Sony was introduced as an add-on to the PlayStation 3 in 2010. The Move uses a video camera in conjunction with motion sensors in a hand-held controller to detect the users position in the 3D space (Kumar 2009). The move works in many ways as the Nintendo Wii except for the use of camera technology, the Move controller has a visible light on the top of the controller which the camera uses to pinpoint the location of the controller. While the Move is not as popular as the two other products, having sold around 15 million units so far (Yin Poole 2012), its use of the camera solution makes it usable for a comparison.

4.2 State of the art

To gather knowledge about the process of developing a gaming device using visual computing, the first logical step is to research the "State of the Art" - I.e. what have already been done, and how it has been done.

As the problem statement of this report revolves around using a webcam to control a game, it is natural to look at the most dominant solutions on the market - i.e. the Microsoft Xbox 360 Kinect, the PlayStation Move and the Nintendo Wii.

The purpose of this research is to gain an understanding of how the team behind the Kinect approached the problem of creating visual computing software, suitable for games, and what they did to solve it - In order to learn from their mistakes and, more importantly, their solutions!

4.2.1 Development of the Kinect

Why the Kinect?

The Microsoft Xbox 360 Kinect is one of the most successful applications of computer-visual interaction made available to regular consumers of recent years, as proved by the sheer number of sold units during the first 60 days (over 8 million, making it the fastest-selling consumer-electronics device in history) (Knies 2011).

As the Kinect is probably the most well-known visual computing device for gaming, and it is the device most closely resembling a possible product described by the problem statement, it seems logical to start by researching this.

Development:

At the onset of development, the Xbox development team were using an algorithm that tracked a user's body movement, and used this string of information to 'predict' the movement. Not only were this method a bit unreliable (it simply couldn't keep up, at times), it was also prone to 'overloading' (Knies 2011) - even if the algorithm could keep up with the speed of the movement, the tracking would become increasingly more inaccurate over time, and eventually crash, requiring a reset, thus making it unusable for extended game play.

Changing the approach:

The team realized, that they had to change their approach, in order for the application to be suited for video gaming. They figured that as the algorithm crumbled due to it trying to interpret movement from a sequence and therefore trying to predict movement, rather than record it, they needed to change their approach to calculating the movement. - *"We couldn't rely exclusively on context, your history, or your motion in the past. "We realized that we had to look at a single image at a time (...) We had to just look at an image and decide what your body pose was. We knew that this was, in theory, possible, because a person can do this. If you look at a photo, you can draw the position of the joints."* - Jamie Shotton, 2011.

So instead of trying to predict something as unpredictable as human movement during games, the team figured that the application needed to interpret movement from every frame of a video stream instead - i.e. they needed an algorithm capable of reading a body's position in space from an image, converting said position into data of movement.

Research in this specific field already existed in the computer-vision literature, which made the approach seem promising.

The initial algorithm attempted to match the entire body at once, comparing the image with an extensive database of possible body poses - in essence, the algorithm recorded the body's pose, compared them with the entire library, until it found the best match.

While this approach kind of worked, it just wasn't going to be efficient enough for the needs of a gaming hardware - As the algorithm checked the entire body at once, it meant that the library had to represent every possible combination of poses, meaning that the library would grow exponentially, causing it to be way too extensive. (Knies 2011) The team looked at the existing literature for answers, and found a way to solve the issue:

They had to break up the body into multiple different focus points - i.e. instead of labeling the entire body, they had to identify and label each joint separately.

Figure 1: Sheep-and-grass example.

The way they did this, was to use the "sheep-and-grass" example: Using an image of a sheep on a grass field, the process isolates the sheep from the grass, automatically labeling the two as sheep and grass, respectively (fig. 1).

This principle was used for the Kinect, to label and color parts of a body. Once this is done, combined with the gray-scale image from the camera, it is possible to locate each joint of the body, as the colored parts are defined as being near a joint. By combining the color code of a joint, and the depth information from the gray-scale image, it is possible to estimate the 3D XYZ coordinates of the joint.

The Kinect comes to life

With the object-recognition approach showing promise, the Xbox team started considering how to implement the algorithm on the Xbox 360 hardware, which, at the time, was already three years old. The team needed to figure out a way to implement the tracking algorithm into the existing hardware, taking the ever more demanding games into account.

To solve this, the team used machine learning - and to do this, they needed data.

"Mat would feed that motion-capture data into a computer-graphics tool that generated depth images so we'd have something to test on where we knew the right answer. We had a ground-truth answer associated with each image." - Fritzgibbon, 2011.

Instead of taking images of real-life people and labeling by hand - a method which is both extremely inefficient and expensive - the team decided to synthesize images, by using computer graphics. After some initial issues with speed or reliability, the team figured out a way to generate the synthesized depth images, which enabled the team to generate millions of images, expanding the training library greatly.

The team then needed to tweak the algorithm to gain greater accuracy, expanding the training library as they went. One problem became apparent, however: it took a very long time to 'train' the program, making the application slow and cumbersome. Using an algorithm dubbed Exemplar (an algorithm used for object removal and labeling in digital images) (Wen-Huang 2005), they solved this issue, enabling the application to work fast and accurately enough to run on every single frame of the data-stream of the Kinect depth camera.

Once this was up and running, the team further tweaked the algorithm, enabling them to turn the application to provide a full skeleton, instead of joints, to provide temporal coherence.

By the time, the algorithm had been completed, it was capable of processing 30 frames per second, using only 10% of the Xbox's processing power - thus making it usable for gaming, as less processing power used by controllers (i.e. the Kinect), means more processing power for the actual games.

summary:

Instead of tracking 'nameless' objects in a stream of images, a reasonable alternative is to record, divide, and label each section of an image - i.e. each joint of a body - keeping track of each part of a body individually, instead of trying to record the body as a whole - this saves processing power, and increases reliability and accuracy.

4.3   Steering Wheel

Ever since the creation of the electro-mechanical arcade games in the late 1960'ies, the control mechanics has seized to mimic the controls of e.g. real life vehicles (Herz 1997). With the joystick, the user is given the feeling of flying e.g. an airplane using the rudder stick, and the same goes for the steering wheel, the first which was introduced in 1974 by Atari for Gran Trak 10 (Kohler 2005). The introduction of the video game console and the personal computer gave competition to the amusement arcades, but opened a separate marked for joysticks, steering wheels and other forms of input devices for home entertainment purposes.

Figure 2: A photo of the Atari Gran Trak 10 from 1974, the first arcade vehicle.

None of the major motion capture game systems (such as the Wii, Playstation Move, Xbox Kinect) comes with a steering-wheel contraption specifically usable for vehicle-oriented games such as racing games. However, there are several examples of console enhancements that will allow the player to incorporate the steering wheel into their game style, enhancing the game-experience and unlock the natural feeling of controlling, most often, a motorized vehicle.

Where the Wii offers a solution that is compatible with the standard controllers, the X-Box 360 Kinect requires the purchase of an additional controller, which is steering-wheel shaped.

Important to note is, that the steering wheel is an independent controller, hence it is does not interact with the Kinect camera. For this particular reason it rather resembles earlier generations of console/PC-steering wheel-controllers, neither which relied on motion tracking by camera.

In between the two is the PlayStation Move Racing Wheel, which is a hybrid of both the Nintendo Wii – and the Kinect steering wheel. Like the Wii, the Move Racing Wheel utilizes the preexisting game-controller, but includes additional functions/relocation of buttons, haptic feedback through vibration, to optimize the game experience.

4.3.1 Steering wheel functionalities

Introduction

To account for the functionalities and usage of the controllers, this section covers the remote functionalities of the Wii controller, the Xbox 360 wheel controller and the PlayStation Move wheel controller. This information will make it possible to delimit the functionalities of the games that are developed to that specific console, and therefore controller, since there can only be as much functionality as the controllers support, in terms of buttons and/or motion control as there is available to the specific controller.
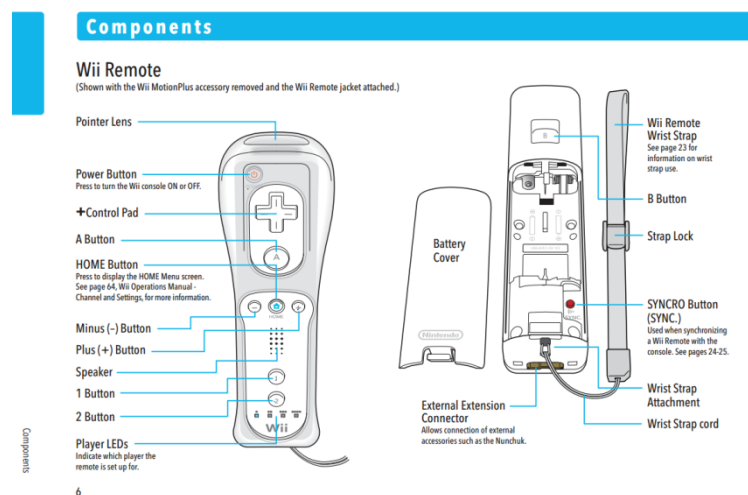

Figure 3: Wii Wheel

Wii Wheel

(Nintendo 2013)
As seen on Figure 3. the Wii remote functionalities are listed in terms of functionalities and components. The Wii remote is mostly controlled through movement and gestures, so there is only a limited amount of buttons. Being:

- Control pad (Up, Down, Left, Right)

- Pointer lens (the method for registering the controller movement)

- A. & B. Button

- Home Button

- Minus, Plus button

- 1. & 2. Button

Figure 4: Xbox wheel controller description (Danish)

Figure 5: Xbox vehicle game controllers

Xbox 360 wheel controllers

(Xbox 2013)
As described on figure 4. The Xbox wireless Speed Wheel contains the following features and functionalities:

- Vibrating feedback

- Movement tracking sensors

- Release buttons for acceleration and braking

- Buttons for game-defined functionalities (Standard Xbox controller functionalities)

  - Buttons: A,B,X,Y

  - Navigation-Button (left, right, up, down)

  - Guide, start, back (console control buttons)

In addition, there are numerous steering wheel controllers compatible with Xbox that utilizes the same as above mentioned functionalities, but also included features such as floor pedals (throttle, brake, clutch) along with gear controller. See figure 5. for a list of examples.

Figure 6: PlayStation Move Racing Wheel functionalities

PlayStation Move wheel controllers

(PlayStation 2013)
As seen on figure 6. The functionalities of the controller are as followed:

- Vibrating feedback

- Movement tracking sensors

- Quick release handlebar levers for motorcycle controls

- Paddle shifters for semi-automatic driving

- Buttons for game-defined functionalities (Standard PlayStation controller functionalities)

  - Buttons: square, triangle, circle, cross

  - Navigation-Button (left,right,up,down)

  - Start & Select (console control buttons)

Summary

This section describes how the steering wheel controller has been developed to resemble a more realistic and intuitive experience when engaging in vehicle based gameplay. There is also accounted for the design in terms of functionalities of the Wii wheel controller, Xbox 360 wheel controller and PlayStation wheel controllers. This section covers the buttons and other controls of each controller to delimit the plausible amount of functionalities that can be implemented in possible gameplay with each controller.

4.4   Game controls in various racing games

Within the State Of The Art of the formulated (IPS) area, we have to find out which choices are popular when it comes to deciding how you should be able to control your racing car in the game. This is needed in order to determine which requirements there are to the design of the controller, so that the user will be able to do the same actions in the game, as he/she would be able to perform in the same game played using e.g. Wii or Kinect.

By looking into which games in the racing category are popular on the three gaming platforms Nintendo Wii, Xbox Kinect and PlayStation Move, we can quickly see a difference in the supplies of racing games for the different platforms. It seems that finding a racing game for PlayStation Move is pretty difficult and the most prominent racing game designed for use with PS Move seems to be the game: **LittleBigPlanet^{TM} Karting** (Miller 2012). Moving on to the Xbox Kinect; here it also seems that there is only one game standing when looking for the most popular racing games, i.e. **Kinect Joy Ride**(Davidson and Searcy 2010). Nintendo Wii, on the other hand, seems to the most popular gaming platform of the three if you want to play a car racing game. For the Wii there are several different racing games that seem to be popular. Amongst those, we have various versions of **Need for Speed**, **TrackMania: Build To Race** and **Mario Kart Wii** (and **Sonic & SEGA All-Stars Racing**, which looks much like Mario Kart in the controls. This will be explained below) (IGN 2013). In order to establish a general rule to follow when you want to create a controller that should be usable with most of the racing games out there, a list of the different controls used by the games need to be made. Making the foundation using the aforementioned 5 games we can assemble the following list:

Consistent for all five games:

- Steering left and right

- Acceleration

- Braking / Reversing

- Changing camera view (at least for the Wii games - different views from game to game)

- Display a pause menu

Need For Speed (Wii):

- Nitro (speed boost)

- Change gear up and down

TrackMania (Wii):

- Braking while accelerating (Drifting in a way)

Mario Kart Wii:

- Choose item

- Throw items back and forward

- Perform mid-air tricks

LittleBigPlanet$^{\text{TM}}$ Karting (Move):

- Use weapons

Kinect Joy Ride (Kinect):

- Charge and release speed boost

- Drift to either side

- Use item

- Perform mid-air tricks in different directions

From this list, we can see that we have some general controls that are used through all of these different games. This means that when designing some form of game controller for a car racing game, you should make the user able to perform the five commands listed in the top, as those commands seem to be important, when they are consistent through multiple games. Furthermore, depending on what game we are talking about, there seems to be about $1 - 4$ additional controls to be able to play the game fully.

4.5   Visual Computing

Visual Computing is computing that lets the user interact with, and/or control work by manipulating visual images (Rouse 2013). The visual images can be photographs, 3D-scenes, video sequences or any other visual output for that matter.

It is the core of controlling and manipulating images and is therefore an important aspect to include in the research and find already established solutions to this method.

One example of Visual Computing is Camspace©. Camspace, is a platform for visual computing that through the use of a standard webcam detects object and hand gestures and then utilize those to be implemented in games, as a mouse controller etc. (Technologies 2009)

It works as a software platform that is installed on any computer with access to a basic webcam, where options to determine the use dictates how the user interacts with it through human gestures with or without an object.

The program can utilize an object as a controller by presenting a solid object with a uniform color that does not match the environment i.e. the shirt that is worn or the wall behind the player. Also, the room must be lit for the color recognition to be possible.

Camspace do offer games and programs that are specifically designed for the software itself, but does also contain drivers for it to be compatible with preexisting games, which is developed with other means of control. Although the games are compatible with the software, it is only a limited amount of controls that can be implemented to function with the program.

Game Title	In-game functionalities
Need for speed underground 2	Move left/right & accelerate/break
Aquadelic	Left/right & horizontal
Trackmania (& Nations)	Speed(depth of object) & left/right
LudoRace	Left/right
OffRoadArena	Speed(depth of object) & left/right
Flight Model Simulator	N/A
Need for speed Carbon	N/A

Table 1: List of vehicle based (driving, sailing, flying) games and the in-game functionalities which is utilized by the Camspace software(Technologies 2009)

Summary

The research indicates that it is possible to create software for computers that utilizes simple webcams for visual computing purposes, such as game control and interaction. It is also possible to modify the use of the program in order to access and control games that is not specifically developed for Camspace itself. Although it is possible, it is only a limited amount of controls that can be imported and utilized via Camspace; (See table 1) such as speed increase/decrease by pulling the designated control object closer or further away, and turn/rotation by rotating the object left or right.

4.6   Image and video processing

Because of the initial naive designs, it was determined that image or video processing would be needed to be researched for finding the position of different elements. In this section, methods for detecting objects in an image are therefore being researched. It is necessary to figure out how to detect certain objects in the scene, to use it for controlling things in games or other interactive media.

4.6.1   Color detection

A very simple and useful way of detecting objects in a scene, is by using color detection. The drawbacks of this are that it is sometimes necessary to put certain colors on objects that will be used in the scene, in order to make them easier to detect. It is also a good idea to have a background with very few colors, so the colors are easier to seperate. It can limit the colors the user can wear, for example if the camera is set to detect green colours and the user is wearing a green shirt. For example, if an image is searched for green colors, and the user is wearing green, the program will always detect the user. While this might be exactly what the program should do, it would be a benefit to be able to control what to detect, and not just give in to randomness.

To enable color detection, thresholding can be used. Thresholding is isolation of a color or a spectrum of colors. For example, to isolate objects that are very red, the statement r > 200 could be used. This would isolate all colors with a red value greater than 200, turning them white in the output image. Here is a simple example on how to use thresholding, written in pseudo-code. This is assuming the input is an RGB image, and the output is a grayscale image.

Algorithm 1 Color thresholding

---

if $R > R_{min}$ and $R < R_{max}$ and
$G > G_{min}$ and $G < G_{max}$ and
$B > B_{min}$ and $B < B_{max}$
 then
   $g(x, y) = 255$
else
   $g(x, y = 0$
end if

In the above example, R, G, and B stand for red, green and blue, and g(x, y) is the output image, where the x and y is the position of the pixel current being processed. For most computers and image acquisition, the RGB color space is used, hence the lack of detail about how to use any other color spaces. In the Gesture Recognition chapter, other color spaces will be discussed, as they make more sense in that situation. The output of the above code would be a binary image. For each pixel in a binary image, it can only have two different values: 0 or 255. This ensures that it is easier for the program to work with, as it only have to compare two values, not 256 values.

Isolating a certain color isn't enough to actually do something with it though. If one wishes to figure out what position a colored object is at, it is a good idea to find the average center of the detected color. A way of doing this by finding the average position of all the white pixels in the output image. So, for each white pixel, add its x and y position to one variable each, and have another variable that counts how many white pixels there are. Then, divide all the x values added together with the number of pixels, and do the same for all the y values.

The computer now knows the average position of a certain color in the scene. But what if the program should detect more colors? One way of doing this, is to avoid using a binary image for the output, and instead have the output be of many different values and or colors. Each thresholded color then has their own color in the output.

4.6.2 BLOB analysis

BLOB stands for Binary Large OBject, and in image processing it represents a group of white or black pixels. As mentioned above, it's possible to detect several colors by having several colors in the output as well. This however means that the program can't actually separate several objects of the same color. To fix this, a BLOB analysis could be used. To separate each BLOB, a so-called grassfire algorithm [1] is usually used, with either 4- or 8-connectivity (Moeslund 2012). 4-connectivity checks the pixels above, below, to the left and to the right of the current pixel, whereas 8-connectivity also checks the pixels diagonally from the current pixel. 8-connectivity is more precise in a sense, since it's able to detect small diagonal lines unlike 4-connectivity, but it takes more processing power.

The recursive grassfire algorithm, which is usually considered the normal grassfire algorithm, "burns" every white pixel in an image, where the white pixels represent something that has been detected. When the program searches through the image and encounters a white pixel, it'll run a function containing the grassfire algorithm. This algorithm changes the white pixel to a gray pixel, with a value of 1-254. By doing this, the pixel is "labeled" with a number from

---

[1] A grassfire algorithm is an algorithm that detects and separates all BLOBs in a scene, by finding a detected pixel, and then labeling all connected pixels as one BLOB (also known as "burning" the pixels).

1-254, so the program can find these labeled pixels later. The reason we can't use numbers below 1, is because that would label the pixel as being black, which the program ignores, and it can't be labeled above 254, because the program detects it as something white which is has to make computations on, creating an infinite loop.

After labeling, the function will check if there is another white pixel next to the one it just marked. If there is, it'll label that pixel with the same label as the one before (it will "burn" the pixel), and check if there are any white pixels next to that one. It's called a recursive function, since it references itself. This continues until the function can't find any more white pixels connected. By doing this the program labels each white "island" of pixels, so it is easier to separate them and analyze each BLOB alone.

Note that there is also a sequential grassfire algorithm, which in some cases is more safe to use than the recursive one, since it doesn't risk running out of memory in the same way as the recursive. The computer only has a limited amount of memory allocated for function calls, and since the recursive function calls itself multiple times, it risks running out of memory if it encounters a very large BLOB, whereas the sequential algorithm doesn't call itself, and as such doesn't risk running out of memory. (Moeslund 2012)

4.6.3   Background Subtraction

The theory behind detecting changes in video is to have a reference image, and subtract that from the recorded image. This effectively removes the background in the image, and leaves only whatever change has been made in the picture. It is also necessary have to perform thresholding, and filter any noise away.

4.6.4   Other detection methods

There exist a lot of other ways to detect something in an image or a video feed, for example it might be necessary to detect how much something is rotated, which in turn could require template matching or other methods to find a certain object with a certain pattern.

It can also be useful to be able to track something in a video feed. Arguably some might say that video tracking has already been described in this research, since detecting a certain color for each frame can be interpreted as "tracking" that color (Moeslund 2012), however that is not necessarily true. In order to actually track something, the program needs to be able to predict where the point it's trying to track will go to in the next frame. Being as many cameras today run at 24-60 frames per second (fps), an object can't move too far in an image unless it's travelling at extremely high speeds.

4.6.5   Summary

Color detection is achieved by using thresholding, which isolates certain colors so they can be detected. Color detection is useful if the program only needs to detect one color, or if it needs to detect more than one easily detectable color (for example detection of red, green and blue, in an otherwise white scene).

However if for example a program is required to detect two different players in a two player game, where each player is holding up a red sticker for detection, the program needs to separate these two colors. This is where BLOB detection comes in use, by detecting each BLOB of white pixels, by using a grassfire algorithm. Note that BLOB detection can be used in turn with almost any other detection method, in order to separate objects in the scene.

If the program needs to detect objects that enter a scene, the program can have a reference image of the scene without any objects, and then subtract that image from the video feed. Anything that enters will then stand out. (Moeslund 2012)

4.7 Gesture Recognition

The concept is focused on using the user as the controller and letting the movements of the user dictate how the computer should react. In order to accomplish this, knowledge of know how to make the computer understand the user' intended action is needed. Since the concept will be camera based, then the tool for reading a user is gesture recognition.

Figure 7: Gestures used to communicate instructions to a helicopter.

What a gesture is and what they mean in relation to the concept.

The common dictionary defines a gesture: *"a movement that communicates a feeling or instruction [...]"* and *"to make a movement with your hands or head in order to show or tell someone something [...]"*. (Macmillan 2005) This means that a general gesture is used as a form of speechless communication between two parties used to relay a message or an intent, as seen in Figure 7 where a movement command is being relayed to a helicopter. Gestures are often used when speech is difficult or impossible.

Because the solution will be based on a camera, then gestures will be the only viable way to communicate an intended action to the computer. Within the scope of the concept, this means that a gesture is a movement, primarily using arms or torso, which communicates an instruction to a computer to be used in an application.

Secondly it is necessary to know how to make a computer understand gestures given to it. For this, Image processing is needed.

The first thing that needs to be done to an image of a gesture is to isolate the gesture from the rest of the image. Some methods of doing this includes background subtraction and hue based extraction based on the hue of human skin (Busaryev and Doolittle n.d.).

Background subtraction (section: 4.6.3) uses the mean values of a series of images of the same background as a basis for subtracting objects from new images. If a pixel in a new image is different enough from the mean value of the series, then the pixel can be used in a binary image of the objects. The problem with this approach is that variable conditions like lighting

and noise can corrupt the results by creating false positives or objects with wrong properties. (Busaryev and Doolittle n.d.)

Another approach is hue based extraction, which converts the image to the HSV color space and then uses thresholding to detect human skin in the image. Caucasian skin has very distinguishable hue values which make it easier to detect through hue rather than regular RGB. (Busaryev and Doolittle n.d.)

A method of recognizing the gesture is the Local Feature Classifier method (Busaryev and Doolittle n.d.) which uses the contours of a hand and points along the contour to distinguish for example one shown finger from two and other gestures. With adjustments, this method can be used on bodies and other features.

Summary

In summary a gesture is a movement used to communicate an instruction to a computer to be used in an application. Background subtraction or hue based subtraction, based on the hue of the skin of the user, can be used to extract a gesture from an image. The Local Feature Classifier method can be used to translate the gesture to an application command.

4.8   User Control Experience

The focus in this project is to make a controller that is based on the user's gestures and movements. In order to accomplish that kind of controller a lot of research is required on how the movements has to be executed in order to make it a smooth and positive experience for the user. If our project wants to achieve that feeling it needs research about how the human body communicates by the use of body language. As body language is a huge subject the group has chosen to only focus on the essentials of the subject and the body parts that we find relevant for our webcam to detect. This decision was made since we only need some parts of this subject and if we should fully understand this subject there should be spend more time than the group can provide for this semester.

In order to create gestures for our webcam to detect there has to be an understanding of what a gesture is. Many have asked this question and according to Adam Kendon an experiment were made with this goal in mind in the year 1978 (Kendon 2004):

"What are the features that an action must have for it to be treated as a gesture?" (Kendon 2004)

This experiment was performed by having twenty people with no psychology or behavioral science background. This background check was made to be sure that it wouldn't have any impact on the test. The people was placed in rooms and told to watch a four minute movie clip without sound made for the experiment. After the movie was done each person was asked to describe what movements they had seen the man in the movie make. The aim was to find out what movements the subjects picked out in their descriptions and to find out what different sorts of movements they identified. After more tests and a lot research the scientists came to a conclusion.

"Gesture is a label for actions that have the features of manifest deliberate expressiveness. They are those actions or those aspects of another's actions that, having these features, tend

to be directly perceived as being under the guidance of the observed person's voluntary control and being done for the purposes of expression rather than in the service of some practical aim." (Kendon 2004)

As recording to the research, gestures have a huge impact on the listeners. Even though webcams don't have any ears these theories can still be of use for us. As Susan Goldin-Meadow writes in her book (Goldin-Meadow 2005) people still use gestures even though they are not communicating with another person. This could be you at home practicing a speech or simple just talking to yourself. You talk but even if you are the only one in the room you would still perform your gestures as you were trying to explain something to another person.

Now that we know what a gesture is the research can continue on which gestures we want to implement in our program. A few common gestures have been found that are being used in our daily lives that could be useable for the webcam to detect (Businessballs n.d.). For the more uncommon gestures the group has to do some testing in order to figure out what would suit the majority of our testers.

Common gestures:
Hands:

- Thumb up - positive approval, agreement, all well

- Thumb down - disapproval, failure

- Index finger and thumb touching at tips - satisfaction

Head:

- Head nodding - agreement

- Head shaking - disagreement

A last thing to mention is that people got a personal space also called Proxemics with five different comfort zones. Close intimate, Intimate, Personal, Social-consultative and Public. All of these zones define how comfortable the person is with another person's standing around them. So this is something that the group should keep in mind when creating our controller. If the aim will be for the Social-consultative and Public zone it is then important according to Edward Twitchell Hall that we place cameras and other players at least 1.2 meters away from the player in order for the person to feel comfortable between themselves and others (Businessballs n.d.).

So what can be concluded from our research is that in order for us to make a good controller that is based on the user's gestures and movements. A test has to be made in order to show how people want to handle the different movements and actions that our webcam will observe. With the information received from the initial tests it is then possible to create a pattern that fits the majority of our volunteers that was used for initial testing. After that it is then possible to implement the right movements and gestures for our controller and then give the right instructions for our movements at the final test. This process should make it a smooth and positive experience to use the controller created for the project.

4.9 Analysis summary

From the analysis, some key elements were found, in regards to the initial problem statement, and how to create a solution for the problem.

The first point that becomes apparent is that while the initial problem statement is technically solved by using Camspace, it also proves to be insufficient for the desired use - As the controls are limited to movement, and not special actions. This is further emphasized as a problem, when looking at the fact, that none of the researched racing games make do with simply movement - they all have one or more additional features, which are essential for the game play.

Another important section of knowledge gained from the analysis, is the different ways to create the program - The analysis shows different methods of doing it, and gives arguments for how each method could be used, which will help deciding, when the product is going to be designed. From the analysis, the initial problem statement can be modified, and turned into a final problem statement, with which to continue work on the project.

References

Busaryev, Oleksiy and John Doolittle. *Gesture Recognition with Applications*.

Businessballs. *Businessballs*. URL: http://www.businessballs.com/body-language.htm.

Castaneda, Karl (2006). *Nintendo and PixArt Team Up*. NintendoWorldReport. URL: http://www.nintendoworldreport.com/news/11557.

Cong, Robert and ryan Winters. *How It Works: Xbox Kinect*. Jameco Electronics. URL: http://www.jameco.com/Jameco/workshop/howitworks/xboxkinect.html.

Davidson, Rob and Matt Searcy (2010). *Xbox Game Manuals*. xbox.com. URL: http://support.xbox.com/en-US/games/xbox-games/xbox-game-manuals.

Goldin-Meadow, Susan (2005). *Hearing Gesture: How Our Hands Help Us Think*. The President and Fellows of Harvard College.

Herz, J.C (1997). *Joystick nation : how computer games ate our quarters, won our hearts, and rewired our minds*. Little, Brown, and Co.

IGN (2013). *Wii U Game Reviews*. IGN.com. URL: http://www.ign.com/games/reviews/wii-u?sortBy=reviewDate&sortOrder=desc&genre=racing.

Kendon, Adam (2004). *Gesture: Visible Action as Utterance*. The press syndicate of the university of cambridge.

Knies, Rob (2011). *Research.Microsoft*. URL: http://research.microsoft.com/en-us/news/features/kinectskeletal-092711.aspx.

Kohler, Chris (2005). *Power-Up: How Japanese Video Games Gave The World An Extra Life*. Pearson Education.

Kumar, Mathew (2009). *Develop 2009: SCEE's Hirani Reveals PS Eye Facial Recognition, Motion Controller Details*. gamasutra.com. URL: http://www.gamasutra.com/php-bin/news_index.php?story=24456.

Lazar, Jonathan et al (2010). *Research Methods in Human-Computer Interaction*. John Wiley & Sons.

Ltd, Nintendo co. (2013). *consolidated_sales_e1306.pdf*. URL: http://www.nintendo.co.jp/ir/library/historical_data/pdf/consolidated_sales_e1306.pdf.

Macmillan (2005). *Macmillan English Dictionary*. Macmillan Publishers Limited.

Microsoft (2013). *Microsoft by the numbers*. URL: http://www.microsoft.com/en-us/news/bythenumbers/index.html.

Miller, Greg (2012). *LittleBigPlanet Karting Review*. IGN.com. URL: http://www.ign.com/videos/2012/11/06/littlebigplanet-karting-review.

Moeslund, Thomas B (2012). *Introduction to Video and Image Processing: Building Real Systems and Applications*. Springer.

Nintendo (2013). *Nintendo*. URL: http://www.nintendo.com/consumer/downloads/WiiOpMn_setup.pdf.

PlayStation (2013). *PlayStation accesories*. URL: http://us.playstation.com/ps3/accessories/playstation-move-racing-wheel-ps3.html.

Rogers, Yvonne, Helen Sharp, and Jenny Preece (2002). *Interaction Design: Beyond Human - Computer Interaction*. John Wiley & Sons, Inc.

Rouse, Margaret (2013). *whatis.techtarget*. whatis.techtarget. URL: http://whatis.techtarget.com/definition/visual-computing.

Technologies, Cam-Trax (2009). *Camspace*. URL: http://www.camspace.com/.

Wen-Huang, Cheng et al (2005). *Robust Algorithm for Exemplar-based Image Painting*.

Xbox (2013). *Xbox Support*. URL: http://support.xbox.com/da-DK/xbox-360/accessories/wireless-speed-wheel.

Yin Poole, Wesley (2012). *PlayStation 3 hits 70 million units shipped worldwide mark six years after launch*. Eurogamer. URL: http://www.eurogamer.net/articles/2012-11-16-playstation-3-hits-70-million-units-shipped-worldwide-mark-six-years-after-launch.

Appendices

A Naïve Designs

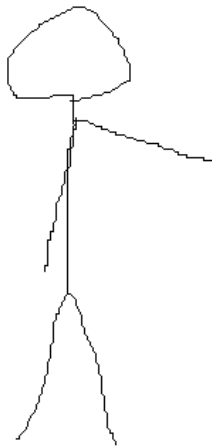### A.1 Control a car through gestures

Naïve Design

Figure 8:

For this design we will use a standard webcam found in most laptops and other types of cameras that can send images to a computer. The idea is to make a vehicle controller, out of a camera, that can react to a user' hand gestures and translate that into input for a vehicle based application or game. These gestures would translate to tasks commonly used in relation to vehicles, like speed up, slow down, change gears, hand breaking. The system should also be used to control the vehicles directions.

The idea here is to use hardware to get input from the user, however this hardware should not be specialized for gesture recognition, like a Kinect or Eyetoy. The hardware should be common to most computer users. The user should only need to install a piece of software and be ready to use gestures.

To make the camera recognize the different gestures some software will have to be implemented. This software will need to recognize the outline of the user and translate the gestures made by the user into commands for an application. The software should be able to recognize

both static and moving gestures, for example, a static, flat hand shown to the camera could translate to a handbrake, or a fast movement of the right arm could translate to a hard turn to the right.

Design Analysis

Motivation
The motivation behind this concept is to replace the standard ways of using controllers with methods that does not require semi-expensive specialized hardware to operate.

This can be used to control applications like driving games without the need for specialized driving wheels, or analog joysticks, requiring only a basic set of hardware already available to most people who own a laptop.

Strengths and weaknesses
Strengths:

- This can make a driving game more approachable.

- This can make a driving game controller less costly.

Weaknesses:

- Precision is very important. We do not want to end up in a ditch.

- Input is not ergonomic.

- Input lag becomes a much bigger concern.

- Input methods are not intuitive.

Target group
The primary target group for this concept is people who are confined to using specialized hardware for certain applications, like driving games or equally like-minded applications.

What do you know / need to know
For this concept a good understanding of image processing is key. The product will be based on image processing to get its readings and translate into input. Precision is key to this product because it will potentially be used for precision operations like navigating a vehicle around a sharp turn.

We need to know how to translate data from a camera into input that the supported applications can understand and use.

What to investigate
The main focus of the investigations is to figure out how much of this is possible and how much of it is possible using a standard webcam. We need to investigate if it is possible to get accurate readings from a camera that is reliable enough to be usable.

Another area of investigation is the industry' perception of the idea. Is it even a good idea? We need to know if the ergonomics of this approach is sustainable.

Initial prototype / mock − up.
An initial prototype could involve using markers on the hands of the user and use one or more cameras to capture data. Software will be needed to translate the input, as well as a demo application involving navigation will be needed. This prototype will be used to answer what is possible with this concept and what needs improving.
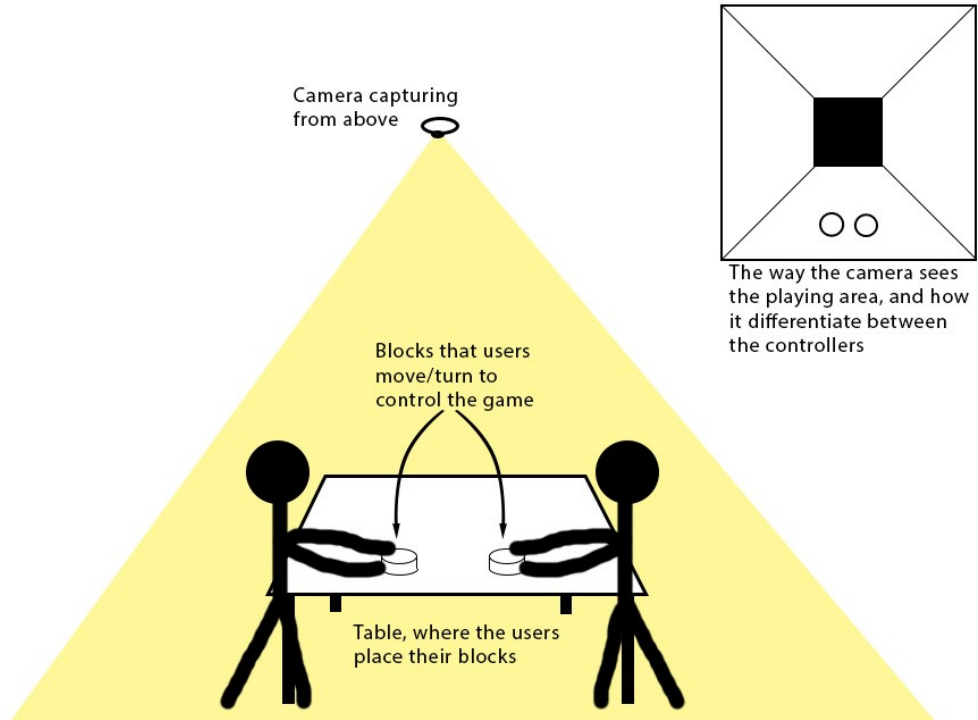
A.2   Controller blocks

Naïve Design

Figure 9: Game controller using blocks. By moving and/or rotating the blocks, the user can control the vehicle in the game.

This idea would give the possibilities to both play as a single player and multiplayer (coop and versus). If we say that we are playing a car racing game, then we would use 2 blocks for each car that has to be controlled in the game. This means that one or two players can control each car. By moving the blocks around, the camera can detect their placement and then use this information to control the car in the game. If we have a table we are playing on, the camera would be looking down at the table from above. In the top right of figure 9 is illustrated how the camera could split up the playing field (where the users are moving the blocks around in) and in this way distinguish between which blocks control what car.

The control blocks would work e.g. as follows:

- Both blocks forward: the car accelerates

- Both blocks back: the car brakes

- Left block forward, right backwards: car turns to the right (and vice versa)

For other games that are not car racing, the blocks could be colored. With this addition, you could also rotate the blocks, which the camera could register by the colors on the blocks.

Design Analysis

Motivation
problem - This idea is based on a way to make a gaming platform, if you will, cheap, compared to other professional and popular platforms such as the Nintendo Wii, and Xbox Kinect, which both tracks movement performed by a user.

Opportunities - The idea here would give the possibility to play together (or against) your friends, as you can play multiple players at the same time. You can cooperate as two (or maybe even more) players in controlling one vehicle in the game, if you would like. This possibility is obtained by using two or more blocks on the board to control each vehicle.

Inspiration - The source of inspiration came partly from the previous idea (Race-making Game) where you are using blocks or something alike, to tell e.g. the car when to turn. An additional source of inspiration was the "music board" where you are using different kinds of blocks, and by placing them on a surface, you create different kinds of sounds.

Strengths and weaknesses
Strengths:

- Options for multiplayer gaming as well as single player

- Cheap: only a webcam is needed, the blocks can be homemade

- The blocks can both be moved around and rotated, giving several ways of controlling the vehicle

- Social gathering, as the idea is to gather more people around the table and play together in the same room.

Weaknesses:

- Camera placement can be a bit problematic: camera has to see the blocks from above

Target groups
As the main problem is focused on making a gaming controller of some kind, so target group will primarily be people already playing video games. This does not necessarily mean hard-core gamers. In fact, non-hard-core gamers would be of preference. Initially this idea might seem more appealing to the same kind of people that uses Nintendo Wii, Xbox Kinect or alike, as this idea involves having people in the same room when playing. Those kind of people might be families playing together once in a while, or friends gathering to play together in another way than LAN parties.

What do you know / need to know more about?
Visual computing - Some knowledge needed to make this idea come true would be how visual computing actually works. The camera in this idea would need to recognize the blocks on the table and read how they are being manipulated to perform an action in the game.

Users - The product of this idea especially has to be appealing to the target group in order to give them the satisfaction they would have gotten from using a popular gaming platform.

This means that the controls have to be easily understandable and manageable as well as many other things, which would need research.

SOTA - At the moment it is unknown if there are ideas like this already developed. So far it is known that there is a device that utilizes something like this idea, but instead of controlling a vehicle in a game, the device provides sound and music according to what kind of block you place on a surface that reads the blocks.

What would you be investigating?
Image analysis and processing - Using a webcam will provide a flow of images of the area in which the users will be controlling their blocks. These images need to be analyzed and processed in order for the software, that will be developed, to understand what the user wants the program to do. To be able to use the images like this research has to be done in the area of image analysis and processing.

Initial prototype
For an initial prototype, it would be useful to focus on using one specific video game. Say we take a car racing game; the prototype would only have to focus on how to control a car in order to see if it is even possible to get near a product where the user would be able to control all kinds of vehicles.

A.3 Immersive Racing Game Experience
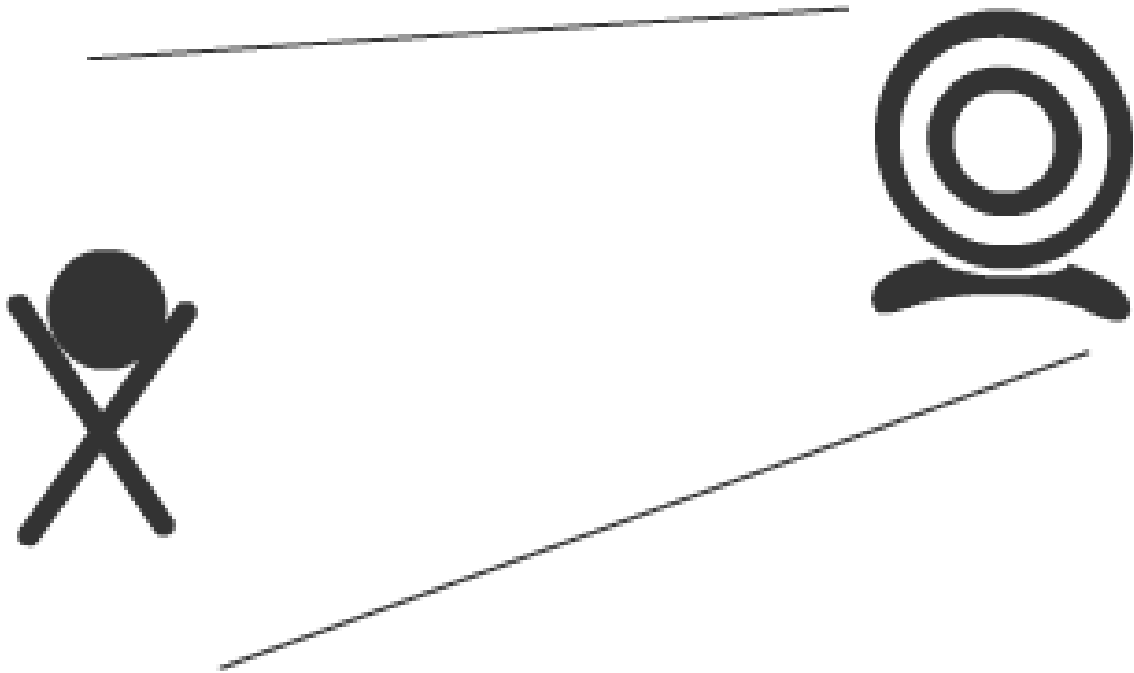
Naïve Design

Figure 10:

The theory behind (IRGE) is to use a static webcam to track your movement/face by recognition and thereby allowing enhanced immersive gameplay by the use of real like movement as required in the specific computer game.

By tracking the whole body you are able to navigate in the game, since we are going to make a vehicle controller based on the user we got many possibilities. We might focus on car games it would be obvious make the webcam track our movements like we were driving a real car. If time is on our side we might also make a controller for other kind of vehicles.

This use of webcam movement tracking can generally be used in games that requires varied movement such as Racing games, First person shooters, puzzle games(Portal) and others, where the player act as a person with movement functionalities within the game itself.

Design Analysis

Motivation

Also as several of the other ideas the motivation for this one is not based on a particular problem that can be solved. Hence it is an idea that is specifically targeted for entertaining purposes only.

The theory is to enhance immersive gameplay by the use of real life movements in the specific computer game, where it is required.

The opportunity with this idea is to enhance the experience by being the subject which is used to control the in-game player in terms of movement.

Strengths and weaknesses

Strengths:

- Could contribute to the already evolving technology of virtual reality.

- Would be a cheaper alternative to more complicated ways of immersion, such as the Wii, provided that the user has enough space.

- Healthiness. The user is able to get some sort of exercise while playing a game.

- Could be a multiplayer experience.

- Has the potential to be fun to make.

- Has the potential to be fun for the users, although that would be subjective.

Weaknesses:

- Requires a large amount of space to interact in.

- Could limit social aspect of games.

- Most people do not have the required equipment and space to make this game fully functional.

Prototype - Mock-up

An early prototype for this concept could be a hallway with a screen and a webcam at the end. In the hallway could be obstacles that the user can hide behind as part of a game. Software will have to make to be made to translate the data from the camera into input to a demo game that we will borrow for the testing.

A prototype would clarify if users would like the idea of a racing game controlled by movements. A prototype could help show the technical limitations of this concept.

Target Groups

The ideal target group for this solution is people who like to play games, but feel that the control options available are insufficient / uninspiring, and would like the games to be more immersive.

What do we know / what do we need to know more about

We know that the concept is somewhat similar to that of the Kinect as far as movement goes. The innovative part of the concept is that the player will be able to see himself on the monitor, and will be able to drive the car around by using steering wheel movements, speed up and down movements and other kind of movements that are needed to control the car or options in the game.

The Kinect is a state of the Art product in this line of gaming, so it will, for obvious reasons, is natural to investigate how it works.

How much research has to be done in this area (from 1-5, where 5 is most)?

- State of the Art: 4

- Full body motion capture: 5

- Perception: 2

- Player motivation: 5

Investigations

For the user to be able to control the game via the webcam, a study upon how you analyze and process an image input from the camera has to be done. The focus here will be how to find and analyze an object/person in the picture and use the person's movements and gestures to apply some action in the game.
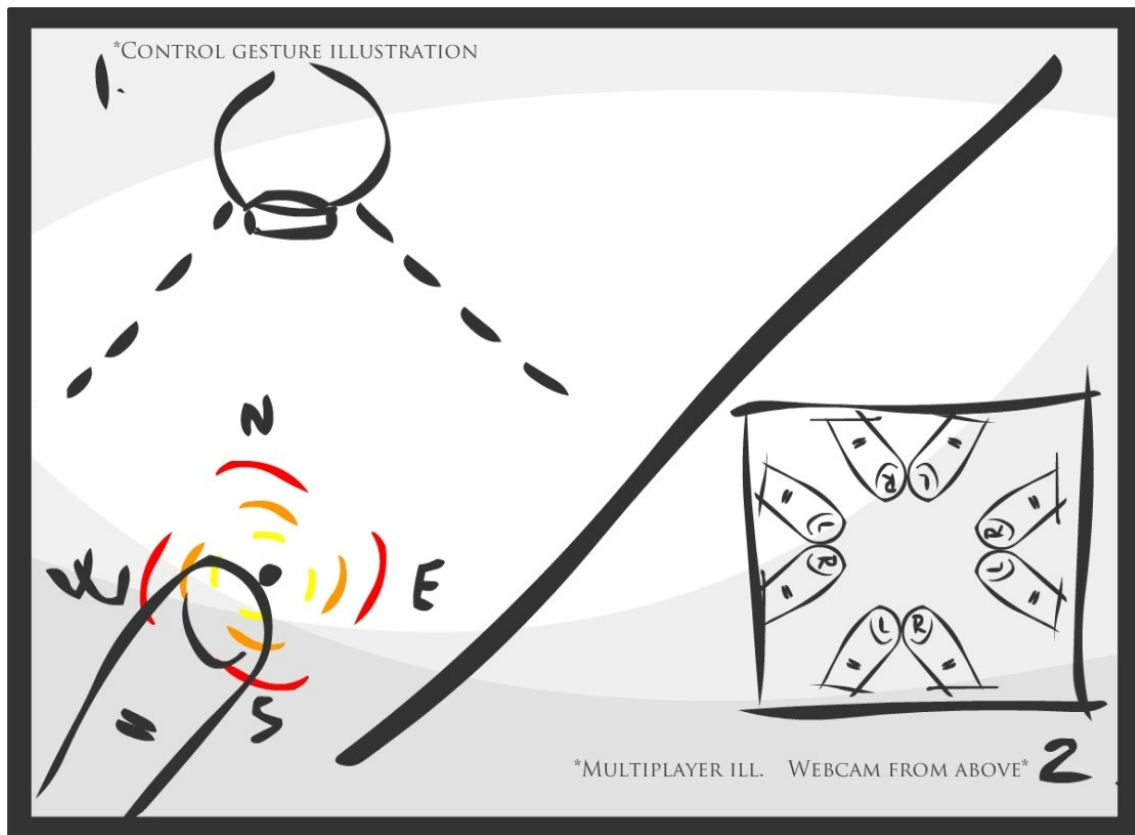
A.4 Pointing controller

Naïve Design

Figure 11:

The idea behind this concept is to use a webcam to track finger(s) on a plain surface to register movement and functionalities of the designated game. By relocating the finger(s), depending on the game, the input will act accordingly to the specific game/functionality.

Also, to incorporate multiplayer functionality, several players can be recognised and because of limited movement to control the specific game, a single webcam would be adequate to keep track of (x) players.

Illustrates the finger movement, and predetermined possible movement/control functionalities of the setup. (Illustration 11: Example 1)

The rotation of the players, can be adjusted down to the individual to address the angle of the screen and the angle of control that the specific player so desires. (Illustration 11: Example 2)

Naïve Design

Opportunities

The idea is based on visual computing where there is no need for controllers. A simple webcam and a plain surface will suffice to create the specified game experience where you control the functionalities with the tip of your finger(s). This method of controlling the vehicle can be modified to combine alternative methods of controlling the game, but also customization of alternative controlling methods possible since there are buttons to limit the possibilities.

Strength/Weakness

The **strength** of the design is first of all the simple hardware requirements which is easy accessible and plausibly with a very simple setup.

Also, by not limiting ourselves to a controller which is already developed with a limited amount of functionalities, we are able to imitate all of the preexisting functionalities and exceed those, since there are no button limits, but rather space limit depending on the single player/multiplayer experience.

The **Weakness** of the design is that by utilizing a camera for tracking movements, you do not get any sort of haptic feedback under any circumstances during the experience.

Also, it is possible that the amount of functionalities that must be present in order to be comparable in terms of functionalities with other game platforms, is too much to cope with, since there are no visual interface lying on the surface while you are playing. Only a visual interpretation of the game mechanics that might appear on the screen.

Target groups

The target group is people whom are interested in playing vehicle games on consoles and plausibly already do so. Preferably we are able to aim at people who are actually playing/performing vehicle gaming on a higher level, in order to research whether or not our take on a plausible alternative to preexisting methods is even a possible solution.

What to know

Theory/Research - First of all we need to figure out what makes a vehicle game the preferred experience with the different controllers in order for us to establish requirements towards, what our product should be able to do and contain.

Furthermore, we need to establish a base of knowledge of the vehicle games in order to figure out which functionalities our product should have. Lastly, we must of course research on how to create such controller for it to be able to function to a plausible large number of games, where the controller would be equally efficient as the preexisting.

Theory/Research
Users - First of all we must establish criteria towards what the users think that the product would need in terms of design and functionality. Also, it would be plausible that the users are so used to "regular" controllers when playing vehicle games, so perhaps, also try a separate

target group of people who do have non to little experience with vehicle games to establish a measurable amount of data of the user experience within both groups.

SOTA - The state of the Art within this matter is the pre-existing controllers on the market. The Kinect controller, the PS controller, Wii and the special developed wheels to emulate the driver experience.

What to investigate

First of all we must investigate the image processing technology behind the idea. What we are able to do and how we can utilize that specific technology.

Next we must investigate the preexisting controllers to delimit ourselves to how we can make the product "comparable with other gaming platforms". What should our product we able to do, can it be implemented, and how can we improve on the matter?

Prototype

The prototype would most likely be an alternative to the pre-existing controllers in terms of key-functionality. We could possibly argue that we have created an alternative method of controlling vehicle games, but since it's a different approach – there would be both advantages and disadvantages which would probably be evaluated individually by the test subjects of the evaluation.

The success criteria would be if the product would suffice when comparing the established requirements and still be functional to a satisfaction of the users, but that cannot be speculated upon before the specific requirements for the success criteria is established.

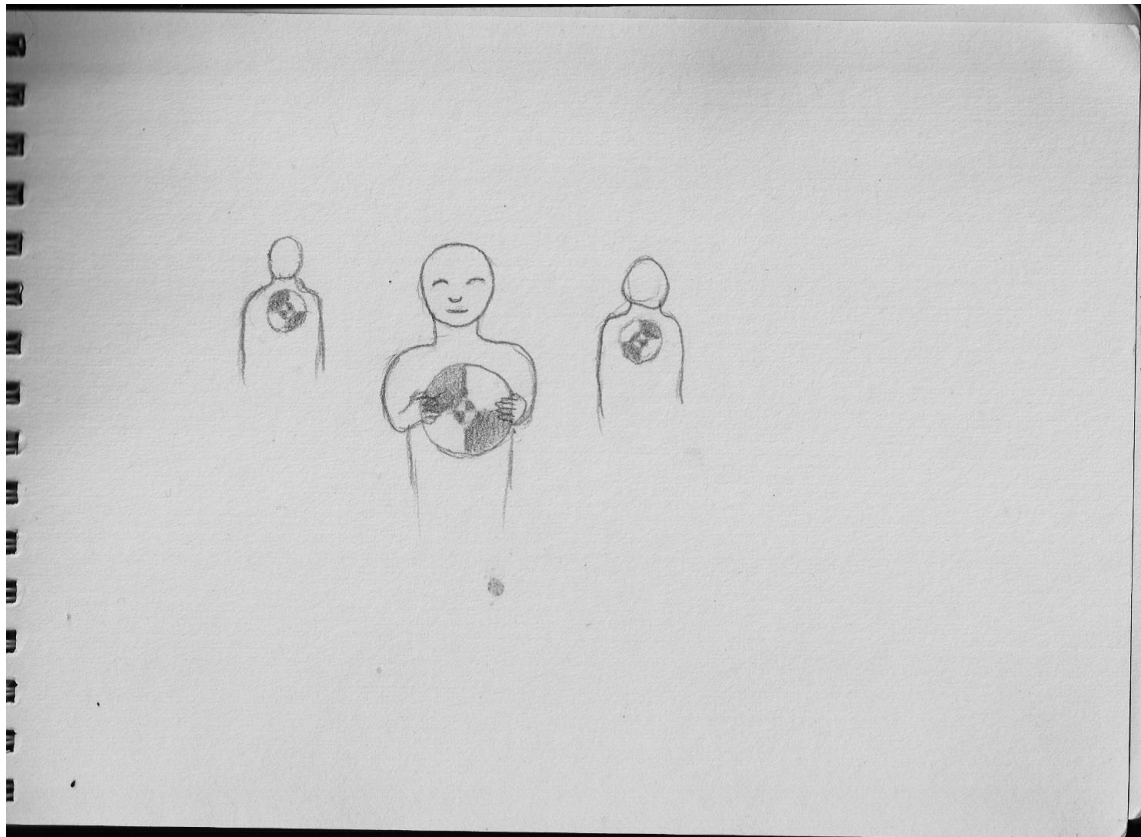## A.5   Object-controlled Visual Processing

Naïve Design

Figure 12:

The idea revolves around a multicolored - or shape-recognizable object that can be moved around in front of the webcam to create a motion capture-device that can be used as a logical vehicle controller e.g. a plate as seen in the image for controlling a car or boat etc.

The objects are going to have some sort of pattern on them to make them easier recognizable for the camera. Due to this pattern, it should be easy for the camera to pick up side-to-side as well as up-down movement, rotation, and forwards and backwards movement. It also makes it easy for several people to use the device at the same time, allowing for potential multiplayer games, or collaborative work.

The currently State-of-the-art products out there such as Kinect, PlayStation Move or Wii only allows for 2-3 players simultaneously. So maybe we will be able to create a cheap alternative to those optional products that are furthermore able to sustain more players. Maybe we could also enable multiplayer online, so people might be able to pick up their plate and join the game with an avatar that reacts to the object's movement.

In order to make this, knowledge about image processing needs to be known. However because the pattern the camera should recognize is a pre-defined static object, it should prove less of a challenge than with a varying object such as a human hand.

Design Analysis

Motivation

The concept behind this idea, is replacing the traditional mouse, with a wireless (and powerless) solution to be able to play vehicle games in an interactive, logical and intuitive fashion.

We are motivated by the fact that products such as the Wii or Kinect seldom can sustain more that 2-3 players, and usually only come with 2 controllers, extra controllers can be purchased, but they usually cost a lot.

Strengths / weaknesses

Strengths:

- Conceptually, the user would be able to use this 'mouse' from a rather great distance (though with diminished accuracy, as a side effect).

- The physical aspect of the mouse could simply be a sticker, or something similar carrying a specific pattern, thus removing the need for a power source.

- The number of players could be limited only by the restrictions of the game.

Weaknesses:

- The accuracy of the device is highly dependent on the skills of the user

- Depending on the quality of the webcam, distance could very quickly become an issue for accuracy, as coding can only do so much.

Target groups

The target group could be anyone who are interested in playing interactive games.

What do you know / what do you need to know?

For this product, a strong knowledge of image processing is key:

A) Knowledge of how to capture, record and convert a specific pattern (like the ones in the illustration) is required, in order for the program to reliably record the movement of that specific object, and not random background objects.

B) Knowledge of converting the data from A) into movement on the computer screen is needed - This could further be expanded upon, to allow for different settings (for different programs).

What would you be investigating?

The main focus would be investigation regarding image capturing and processing.

Additionally we would need to gather some information regarding how web cams work, how the different types differ from one-another, and how this will affect the program - And how (if possible) to counteract this.

Networking would also be required to investigate if we were to implement an online solution.

What could be an initial prototype / mock-up?

An initial prototype / mock-up are not easy to make, as the solution is rather simple. One solution, however, could be to create a functional program, with only limited actions available (such as simply moving the mouse around, using image capturing.
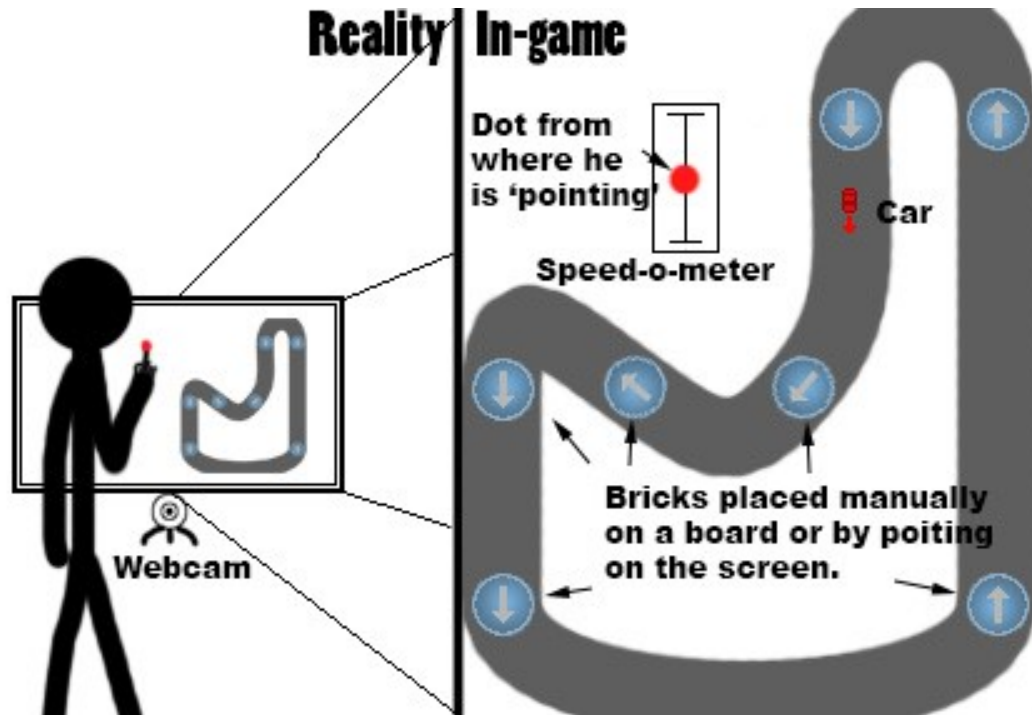
A.6 Race-making game

Naïve Design

Figure 13:

This idea is inspired by some of the racing tracks made for kids to build. You put the track together, and then you had a controller that could only control the speed of your car and nothing else. The goal was then to try and keep your car on the track, while still going at a maximum speed.

Of course what we should be explaining here is the controller, not the game itself. The controller has a board (not in the above picture), where you place down the bricks as "points" where you would like the track to go. The game then connects these points with lines, and there you have the track! This would use image processing to figure out where the bricks were placed.

You then use your finger to control the speed of the car, trying to make it go fast, but not too fast. You need to put a colored thing on your finger so the camera can see where it is placed.

Note that this idea could be combines with some of the next ideas explained, and the control schemes could be altered to give you more control of the car.

Design Analysis

Motivation

This idea has roots in our childhood, and because of that it is something that we can relate to.

The game itself isn't too complicated, with the hardest part probably being how to connect the dots to form a track, and making the car controls feel nice. The image processing part of it should be fairly simple.

This idea could also potentially require us to build a table of some sort for the bricks to be placed on. This means that there will also be some work in creating something physical, which some group members might like.

Strengths/weaknesses

Strengths:

- The idea is something we can relate to.

- Could be cheaper than buying an actual racing track.

- Allows users to be creative by being able to create their own things with the provided material.

- The idea itself is fairly simple.

- Allows for social interaction.

Weaknesses:

- Could potentially require 2 cameras, however this can be fixed by using the board where you place the bricks as the "control-space" as well.

- We're not sure yet exactly how we can give the user the ability to create the track themselves yet.

- Requires quite a bit of programming, might not be fun for all members.

Target group

The target group would be anyone who plays games. Younger people might be a greater target group however, since they might've played the physical version of the game, and has some nostalgia when it comes to the game.

The game should be really simple and intuitive to use, so you don't need a lot of technological expertise to use the product.

Knowledge

We know for a fact that some students have made tables that are able to detect objects on top of it before, so some research has already been done in this area, and we'll have some other products to compare ours to. The same has been done with color recognition.

In order to create this idea, we would need knowledge about image processing to create the controlling features. We would also need knowledge about how to create the actual game. This could be done in a lot of programs such as Unity, Flash, Processing and more. We could potentially also create it from scratch using c++, however this would most likely not be preferred.

We also need knowledge about vectors and vector calculation in order to create the track, and some knowledge about 2d physics, to make the car act in a way we want. However a program such as Unity already has a physics engine, potentially saving us time.

We can make loads of prototypes of this idea, and it has a lot of "steps" on which we can test the game. We can test the initial color recognition without the need of anything else, we can test the creation of the track without the need of anything else, and we can test the car 'mechanics' in a way without the need of much, except a track to follow.

A.7 Tabletize your computer!

Naïve Design

The idea for this design is, using a standard webcam, simulating the rotation measuring devices of modern tablets and smart phones, allowing computer users the same intuitive means of controlling games.

As the webcam is recording, edges will be recognized (using image processing), and the movement of said edges will be converted into movement on the computer - In other words, movement within the captured pictures will be replacing the tilting sensors of the tablets and smart phones.

Design Analysis

Motivation

The motivation behind this concept, is to create software, that will allow a webcam to replicate the tilting and rotation sensors ordinarily found in e.g. tablets and smartphones. This would allow desktop and laptop computer users to immerse them selves in similar games - ie. games where the controls are more intuitive than simply using the WASD movement keys, for example.

Strengths and weaknesses

Strengths:

- Allows for a more immersive experience

- Expands the range of games available to computer users even further.

- Could possibly be utilized for other purposes, where a tilt sensor could be useful.

Weaknesses:

- Hardware not built for this kind of action, unless using an external webcam.

- Potentially hard / impossible to do general calibration, causing the device to needing recalibration after either every game session or after every time it is moved from location to location.

- Unless using external webcam, the setup can be both heavy and cumbersome, making the device less intuitive than the alternative, thus defeating the point.

Target group

What do we know / what do we need to know more about:
We will most likely be using openFrameworks (C++ coding) to process the images into movement data. What we need to know is, how to convert the stream of images from the webcam into usable movement data (ie. how to detect the movement in the images reliably). Furthermore we need to convert this data into actual movement in a game.

Investigation

We will need to investigate similar, professional, solutions - sort of like the tablets and smartphones, but we should try to see if we can find someone who have done the same - without the use of tilt-sensors.

We will need to research a lot of image processing, and figure out how we're going to approach the movement - record - movement process (ie. recorded movement -> data -> output movement), and how much movement should account for how much movement on screen.

Initial prototype

An initial prototype is not easy to do, as it is highly software based, and thus will more or less be finished, once an initial prototype is ready for testing. "Simulations" (or Wizard of Oz-experimentations) could be used, however, as this would give us some idea of how the program will work and feel for the users, while being very time efficient.