Wii Remote Interaction for Industrial Use

Marcus Nielsen, Michael Stenbacka

November 16, 2009

Chapter 1

Abstract

By focusing on the potential of the Wii Remote, we have implemented a broad spectrum of concept ideas into the same package, in an effort to give a good overview of the Wii Remote's properties such as mobility, direct manipulation and generally high affordance. The purpose of this work was to find a concept on how a Wii Remote can be used as a tool for the industry, outside the domains of gaming and entertainment. The environment for our investigation was ABB's Robot Studio which is a simulation tool for industrial robots. Creating a concept with today's products enabled us to discuss a present solution and also a possible future in form of a redesign rationale that we exemplified with a set of scenarios.

Contents

Chapter 2

Introduction

2.1 Problem Description

The industry today uses robots to automate production and to keep humans away from health hazardous workspaces. When a costumer has bought a robot from ABB Robotics they need to program it themselves. Robot tasks are programmed by using a tool called Flex Pendant. This tool is connected to a controller that handles robot tasks and related actions. The Flex Pendant, as well as new tools, is continuously being researched to increase efficiency, lower the cost and increase the usability.

2.1.1 Scenario

Erik and Peter, two loyal customers to ABB, are standing inside a dirty industrial building and programming two ABB robots. Erik, owner of Aros Weld and expert in sheet metal welding is trying to convince his colleague Peter how to solve their problem of welding two small metal pipes. Peter, a master of robot programming dislikes most of the creative ideas streaming out of Erik, because he wants his robots to act like human welders.

A lot of miscommunication arises since Erik is an old Swedish weld expert with no academic experience and Peter is an engineer from USA. We call this the problem of **communication**.

While compromising and arguing, using pen and paper, they draw models of how the two robots should behave. But it's finally up to Peter to actually create paths and tasks. After some hours of debating and planning they finally come up with a solution for the welding order of the week.

Erik calls his customer to discuss the price but unfortunately he does not notice the thick wire from the Flex Pendant. Erik trips over the cable and drops his cell phone in the ground while Peter yells:

- "No! Catch the Flex Pendant!" - "Why didn't you put that thing somewhere safer?! Erik yells!" - "The cable was too thick to put the pedant in the locker. It's not my fault, Peter yells!"

A cell phone may not be the most expensive thing but a Flex Pendant, that's a whole different story. This leads to the problem of **flexibility**.

After installing a new Flex Pendant from ABB Robotics, since the last one fell down during the accident, Peter asks Erik if he can take a trip back to London to celebrate his honeymoon. A worried Erik asks Peter: "Can't you wait? We got more components arriving next week." But Peter thinks that Erik should be able to fix the small adjustments needed to weld the parts by himself. Erik tries to jog the robot, using the Flex Pendant, but fails and angrily walks into his office and calls the product manager of the Flex Pendant: - "Can't you, as leaders in the robot industry, create tools easy enough for us normal people to use and without the big stupid cables!?"

The experience and knowledge needed to jog robots fast enough to be productive leads to the problem of **expertise**.

2.1.2 Summary

From this scenario we could identify three problems of robot programming today. The requirement of both skill and professional experience. To handle expensive tools that require cables to be connected to it and therefore diminishing mobility. The lack of multi-interaction, so more then one engineerer could work synchronized.

2.1.3 Wii Remote for Industrial Use

The purpose of this thesis work was to investigate how the Wii Remote could be used as a tool for the industry, meaning outside the domains of gaming and entertainment. The goal is not to replace mouse and keyboard in areas where they are already well established and our solution is not meant to be a full system. Instead we focused on creating a concept demo, showing how this new technology could be used in the domains of industrial applications.

We base our work on different types of gestures, since the Wii Remote doesn't have a keyboard. With only a few buttons a user should be able to perform a large variety of actions inside RS2008 (RobotStudio 2008, see chapter 2.4). The concept demo had some basic requirements, it should run smoothly without crashing, be easy to use and show functionality based on the possibilities of the Wii Remote.

2.2 Overview

This report has been split into 7 chapters starting with this introduction on the Wii Remote and on applications we have used in our project. In chapter 3 we describe the theoretical essence that is fundamental ground for our work. This is coupled with our own experience in chapter 4 where our implementation is described. Chapter 5 show the results from our implementation with some user feedback from a user test at Robotics partner seminar 2008. Spin-offs and ideas of future work is described in chapter 7 showing how a redesigned implementation could look like. A discussion on this redesign with related theory is presented in chapter 6. This is summarized in chapter 8 with our conclusions.

2.3 The Wii Remote

On November 19, 2006 Nintendo released their seventh generation gaming console called Nintendo Wii. Together with this new gaming console, a new kind of game controller device was also introduced. This controller device, called Wii Remote and also known as the Wiimote, is a wireless controller equipped with an accelerometer and one IR-camera (see figure 2.1). On the outside it has 6 buttons and one traditional arrow button. To give users feedback it is equipped with blue leds, a rumble device and a little speaker. The Wii Remote can also be extended with more devices as the Nunchuck. This device is optional but has the same accelerometer features as the Wii remote and is equipped with two trigger buttons and one analog stick. The

Nunchuk is held in the left hand and is connected by a chord to the Wii Remote.

Figure 2.1: Wii Remote

The Wii Remote sends data wireless with a Bluetooth connector/receiver to the Wii console. To be able to get this data transferred into a PC we connect the Wii Remote using a normal Bluetooth dongle.

Referenced material of the Wii Remote has been extracted from [1] since there is no official detailed specification available.

2.3.1 The Accelerometer

The accelerometer inside the Wii Remote and Nunchuk extension is a ADXL330 accelerometer [21]. It can register acceleration in three dimensions. When holding the Wii remote still it has continues force on its Y axis since the gravity always expose force on the accelerometer. When tilting the Wii remote we expose different axes to this gravity force and could thereby know the angle of the Wii Remote relative to the ground. Since the Wii Remote can only feel its angle relative to the direction of the gravity, our orientation will lie in a two dimensional circle. This means that when we rotate the Wii Remote perpendicular to the ground, we will not get any clear readings of our new state.

2.3.2 The IR-Camera

The possibility to aim at a certain point on a screen was very limited using accelerometer values only. This due to the above noted problem of rotating perpendicular to the gravity field. Nintendo solved this issue by adding an optical camera on the Wii remote and included a bar with some IR-LEDs

---

[1] http://www.wiili.com/index.php/Main_Page

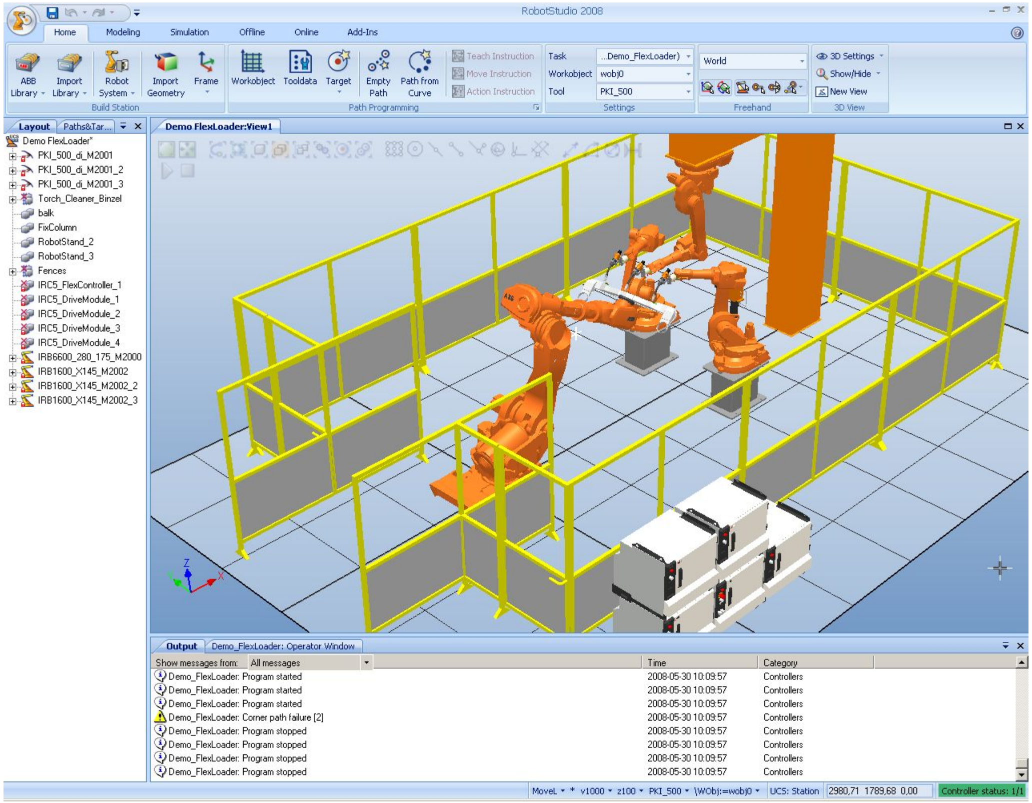
mounted. This bar was named the ”sensor bar”, but should not be confused with an actual sensor since it only emits IR-light. By aiming the camera toward the sensor bar we could pinpoint the position at which we were pointing at relative to the sensor bar. The camera can register four sources of IR-lights, but Nintendo only use two since it is enough for calculating distance and to keep track of its own rotation.

2.4 RobotStudio

RobotStudio 2008(RS2008) is a software made by ABB Robotics [2]. It is a CAD (computer aided design) tool for offline robot programming with the goal to minimize production downtime. RS2008 give robot engineers a 3D environment to create realistic robot simulations and workflows (see figure 2.2). All Robots, also known as ”mechanisms” in RS2008, are coupled to a virtual controller that simulates the real controller used for a real ABB robot. In RS2008 you find most of the functions viable in common CAD programs today, different ways of creating, changing and viewing objects in a 3D space. You can also find functions for robot programming like freehand jogging, creating paths and run simulations.

---

[2] http://www.abb.se/product/se/9AAC910011.aspx



Task ...Demo_FlexLoader)
Workobject wobj0
Tool PKI_500

Show messages from: All messages

```
                                                      Time                 Category
Demo_FlexLoader: Program started                      2008-05-30 10:09:57  Controllers
Demo_FlexLoader: Program started                      2008-05-30 10:09:57  Controllers
Demo_FlexLoader: Program started                      2008-05-30 10:09:57  Controllers
Demo_FlexLoader: Corner path failure [2]              2008-05-30 10:09:57  Controllers
Demo_FlexLoader: Program stopped                      2008-05-30 10:09:57  Controllers
Demo_FlexLoader: Program stopped                      2008-05-30 10:09:57  Controllers
Demo_FlexLoader: Program stopped                      2008-05-30 10:09:57  Controllers
Demo_FlexLoader: Program stopped                      2008-05-30 10:09:57  Controllers
```

MoveL * v1000 z100 PKI_500 \WObj:=wobj0 UCS: Station 2980,71 1789,68 0,00

Figure 2.2: RobotStudio 2008

Chapter 3

Background

In this chapter we provide knowledge valid for understanding our project, the terms we use and also the related work. This includes new interacting techniques, with focus on 3D motion sensing devices, designing interactive systems and gestures recognition.

3.1 Gestures

We will base our work in the domains of gestures. Gestures could be grouped into deictic, gesticulation, manipulation, semaphores or sign language. Of these five styles, we will work with deictic, manipulative and semaphorical gestures [7].

The definition of the word deictic is "Linguistics of or relating to a word, the determination of whose referent is dependent on the context in which it is said or written. In the sentence I want him to come here now, the words I, here, him, and now are deictic because the determination of their referents depends on who says that sentence, and where, when, and of whom it is said."[1]. In the context of gestures, we might want to drag and drop a marked object on top of another. If we drag-and-drop a music file onto a music player, we might add that file to a playlist. The deictic gesture here is the actual drag-and-drop, while adding the music file to the playlist is just a context sensitive effect of this action. Manipulative gestures are those "whose intended purpose is to control some entity by applying a tight

---

[1] http://www.thefreedictionary.com/deictic

relationship between the actual movements of the gesturing hand/arm with the entity being manipulated.” [7]. Both the Wiimote’s accelerometer and IR-camera register movements in a way that enables manipulative gesturing. Translating the position of an object in a 3D world, by moving the Wii Remote or Nunchuk, is one way of using manipulative gesturing.

A semaphore gesture is what most of us relate to when we see the word gesture. It can be either a static pose, like ”thumbs up” or a dynamic one, like tapping your foot to illustrate impatiens. The gesture technology can be perceptual or non-perceptual depending on if it needs to sense the user’s input, in which case we have a perceptual technology, or if we take the input directly from a device as our Wiimote. Hence our Wii Remote is of the non-perceptual nature, but can be used as a perceptual technology if we use the camera to recognize our fingers move as Johnny Lee has shown in one of his inventions [9].

3.2 Human Computer Interaction

As shown in the introduction scenario, the design of the interaction between a human and its system is of a great importance. The interface should be user friendly and this could be achieved by studying HCI methods, Human Computer Interaction. Human Computer Interaction, is a discipline were the interaction between people and computers are being studied. This includes design, evaluation and implementation of interactive computing systems for human use [6].

Computer science and applied psychology are blended to provide knowledge and methods for designing user friendly environments. The central terms in a user friendly environment is usability, utility and relevance [10]. The interface between a system and a user is said to be well designed if it has a good usability, which basically means that a system behaves as a user thought it would [18]. To achieve this, a system should be easy to use, be safe and efficient. By safe we mean it handles all the unexpected interaction and prevents the user to perform faulty actions. By efficient we mean it should take less time for a user to find the right buttons, tools and functions. Although, the interface should not be compromised in such a way that it have less functions or abilities necessary for its task. As users ”never come with a blank state”, research about who they are and how they act is valid to form a relevant system with high utility.

There are 12 principles that provides a guideline for interactive systems design. These principles are: Visibility, consistency, familiarity, affordance, navigation, control, feedback, recovery, constrains, flexibility, style and conviviality [1].

3.2.1 Prototyping and evaluation

Two other important terms in a design process is prototyping and evaluation. With prototyping a designer could create a simplified version of the full design in an early stage, test it and then throw it away. This prototype does not even have to be computer based as it can be made on a simple paper. There are of course more sophisticated types of prototyping evolving special software.

By evaluation we mean testing or trying. This could be done as early as trying out a conceptual idea with a scenario or inviting users to try the present system during the design/implementation process. It's a necessary task to evaluate if your current design has a high amount of usability, utility and relevance.

3.2.2 Scenarios

Using scenario based design is one way of identifying who and how users will interact with a new system. This is also an essential part of prototyping and evaluation. A scenario is a short story about people and their activities [3], i.e. we used a starting scenario in this report to backup the reason for our project. By using scenarios, it becomes easier to identify problems with a new design and it also provides a tool for expressing a new design for people outside the project.

3.3 Gesture Based Systems

Gesture based interaction has been around for several decades and got more commercial popularity with Nintendo's console release [7]. With this console a new device called Wii Remote was released. It was relative cheap and introduced a new way of playing console games. Unlike old interaction methods, using buttons and sticks, the Wii remote use acceleration to create game play, i.e. golf and tennis.

Researchers at the VTT Technical Research centre in Finland have been working on gesture recognition with a device called SoapBox [11]. With the SoapBox prototype they controlled a DVD player with basic gestures. They could boost the system so it had an average accuracy rating of 98 percent from a set of 240 gestures.

3.4 Gesture Recognition

Gesture recognition is a research close related to our own work. A general pipeline that is common for gesture recognition is shown in figure (3.1) and there can be variations of this pipeline [16] [11].

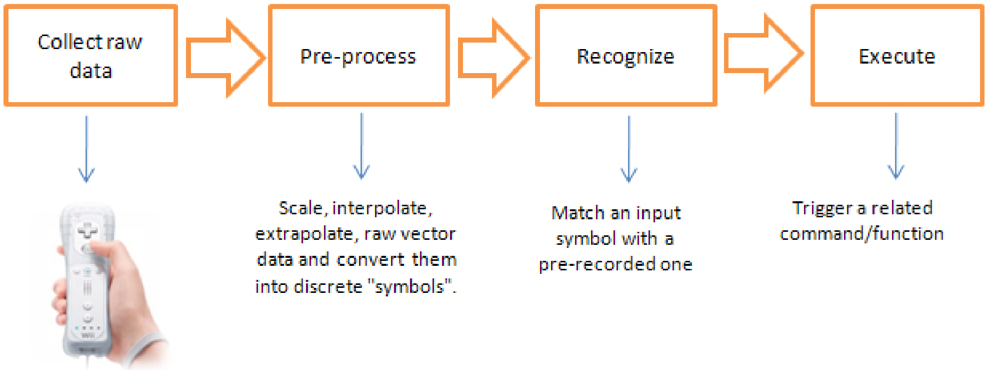

Collect raw data
Pre-process
Recognize
Execute
Scale, interpolate, extrapolate, raw vector data and convert them into discrete "symbols".
Match an input symbol with a pre-recorded one
Trigger a related command/function

Figure 3.1: Gesture pipeline

The first task in this pipeline is to collect raw un-calibrated motion data, i.e. read 3D motion data from the accelerometer.

The second task is to pre-process the raw vector data. The first step in the pre-process is usually to scale, interpolate or extrapolate the data and sometimes reduce or generate noise [11]. The next step in the pre-process stage is vector quantization. The three dimensional vector is converted into one dimensional symbol. This is needed to reduce the amount of data without losing any valid information. A common way is to cluster all the raw vectors with a k-mean algorithm [16].

The third task in the pipeline is to recognize a pre-recorded gesture. There are several techniques that solve this problem. Both soft computing and more sophisticated methods, such as HMM, Kalman filtering are common strategies.

Hidden Markov Models, HMM, is a well known strategy for motion based recognition and is well established in speech-recognition [14]. HMM is a stochastic process that is built on Markov chains [12]. A Markov chain is a way of handling states, given a finite number of states. The future state is independent on past states, so all information needed for future processing is included in the present state only. Being a stochastic process means that a transition between states is based on a probabilistic. The reason why it's called hidden is because the model hides the real states and only show the associated observation symbols. The researchers at the VTT Technical Research Center describes that the actual gesture recognition is to find "an index of the discrete HMM which produces the maximum probability of the observation symbol sequences" [11].

There are other techniques for gesture recognition other than pure motion. For example, vision based tools that recognize objects and patterns are used by a mobile robot for terrain navigation [13]. There are also image-processing techniques that detect shapes, textures and colours for gesture recognition.

3.5 Wiimote-Based Gesture Recognition

Schlömer et al. created a gesture recognition system which used the Wii Remote as an input device and HMM as an model for the recognition part [16]. Their concluding recognition results vary between 85 to 95 percent. Worth noticing is that the simple roll gesture had lowest result with 84.3 percent while the "tennis"gesture swinged up to 94.5 percent. Since using the roll gesture as a semaphore filters away the speed of the roll as well as how far the roll went, we can note that some gestures are better fitted as strictly manipulative instead of semaphorical. Their gesture recognition library is available to the public for further research.

Shiratori and Hodgins have implemented an accelerometer-based user interface to control a physically simulated character [17]. With two or three Wiimotes and the use of Kalman filtering, they simulate actions such as running, jumping, and turning.

Lee, Kim et al. uses the Wii Remote to create collaborative art with the use of the Wii Remote's infra-red camera [8]. The main purpose lies in the fact that this study has worked with "the user's 'reflexive' awareness towards their own body" which means that they worked towards a transparent controlling interface which has high affordance.

Guo and Sharlin makes a comparative study between the Wii Remote as a general tangible user interface and the keyboard [5]. Although the results suffer somewhat from a non-optimized keyboard scheme, the results points towards the possibilities of an effective, non-discrete posture controller. The learning curve seemed to vary with the conclusion that a higher "degree of integration" and "degree of compability" of the tangible user interface and the robot meant that beginners did not do as many faults. One gesture mapping scheme was based on the analogy of horseback riding, but in real-time, the user still needs to recall rather than recognize [23]. This can lead to erroneous output especially when the user needs to act fast or continuously.

3.6 3D Navigation

Navigation tools in three dimensional (3D) based applications is something familiar for those working with 3D CAD, 3D studio max [20] and ABB's Robot Studio. But for some users it can be very difficult to learn [4], resulting in users rejecting 3D tools and continuing to work with 2D even thou a 3D application would provide more utility, if mastered correctly.

3D navigation involves the repositioning and reorientation of a view, often analogized with a camera. The most common tools needed for navigating a 3D camera are pan, zoom and orbit [19]. Pan will only change the position of the camera without affecting the orientation of it. Zoom will move the camera closer to a certain focus point, most often a selected point or an object. Orbit enables us to move around a focus point while continuously keeping the camera directed towards the same focus point.

Issues in 3D navigation involve disorientation due to an empty view or the lack of transition animations between different views as well as users inability to choose the right tools appropriate to navigate efficiently. New 3D users try to use the tools with the skills and knowledge derived from their work with 2D tools [4]. Fitzmaurice et al. have also found that users have a strong idea of how a tool should work without knowing about the real terminology for 3D [4].

3.7 Augmented Reality

Augmented reality is a combination between the real- and the virtual world. It is interactive in real-time as well as being three dimensional [22]. Augmented reality is a sub domain of virtual reality and focuses heavily on enhancing real objects digitally. Augmented reality still has technological and ergonomic issues, but certain areas such as "Spatial Augmented Reality" has gained in interest due to a fall in cost and increased availability [2]. Spatial augmented reality is famous for its hologram concept as seen in science fiction movies, but also for transparent screens that can mark moving objects behind the screen or show some details about the moving object and is of use for the military pilots and soldiers as well as civil drivers.

The application domains that are mentioned by Bimber and Raskar are of importance to us. It ranges from mobile and ubiquitous to virtual- and augmented reality[2]. Mobile applications are a well known fact due to the everyday mobile phones and will not be further explained. Ubiquitous applications tend to be small, cheap and integrated in our everyday environment [24]. The user may thus interact with a system without its own explicit knowledge. Smart fridges are a good example, where it can keep track of food and give possible menus or a shopping list.

An augmented reality display needs to intersect the optical path from the real object and the user's eye to be able to modify the visual content received by our eyes. Typical displays can be planar, as a normal desktop flat-screen, or non-planar, as a hologram would be. Depending on how the display is placed, different technological implications are bound to the display. We can group the displays into the following categories: Retinal-, Head-Mounted-, Hand-Held- and Spatial displays.

Retinal displays send the augmented view directly onto the retina of the eye. In the future this can enable high resolutions and large field-of-views. Note that today, only monochrome, non-stereoscopic versions exist. Head-Mounted displays can be either video see-through, where a camera input mixes with the augmented layer and then sent as a complete view, or optical see-through, where different transparent materials that can display the augmented layer while letting through normal light from behind. This kind of technology has issues concerning ergonomics and picture quality [2].

Hand-Held displays range from today's PDAs and cell-phones to more specialized tools. These units have limited field-of-view and processing power. Bimbar and Raskar points out that moving a display over a static scene can effectively count as a "large" view than a static display of similar size where the scene is moved [2]. The broad market and game industry drives the mobile products forward, and makes the hand-held alternative viable for augmented reality.

Spatial displays are divided into video see-through, optical see-through and direct augmentation. Both methods for the "see-through" has obvious issues of field-of-view. Video see-through has general problems with merging the less quality of real world objects with the virtual, augmented ones. Optical see-through displays are generally less stressful on the eyes, has higher resolution, large field-of-views and scales with the environment[2].

Projection-based spatial displays have many domain-specific issues including shadow casting of real objects, no possibility to project virtual objects in free space and non-zero parallax removing multiple user possibilities. A good thing about projections is their close bounds to the real world as well as the ergonomically benefits and the scalability. As a whole, this kind of technology enables available and intuitive interfaces[2].

More about augmented reality will be discussed in our design rationale were Augmented reality could be mixed with Wii Remote interaction to create new possible ways of interaction and visualization.

3.8 Robot Jogging

An industrial robot can be programmed to perform preprogrammed operations in a working area. To program an ABB robot, you can manually write code on how the robot should move, or jog a robot with a Flex Pendant (see figure 3.2). Jogging the robot means that you use an analog stick to move the robot arm to the desired positions with different types of path settings.

There are three types of jogging capabilities in RS2008, jog linear, jog joint and jog reorient. Linear jogging is kinematic motion of the whole robot were the flange is moved along a linear path. The flange is the endpoint of the robot where a tool can be mounted. Joint wise jogging is an operation which can rotate one joint. With Jog reorient it's possible to change the angle flange without changes it's coordinate.

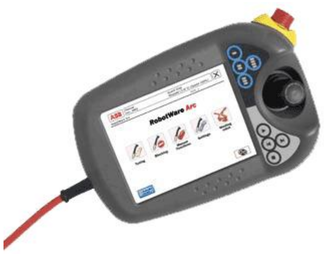


Figure 3.2: Flex Pendant

Chapter 4

Design

To be able to test Wii remote interaction techniques in the domains of ABB, RS2008 was choosen as the pilot environment. It was a perfect match, since it simulates a 3D world and robot programming.

4.1 Creating a Concept

To familiarize ourselves with RS2008 and robot programming we went down to Gothenburg to meet the developers of RS2008. After a workshop with the developers and a interaction designer we ended up with two main design goals: robot jogging and camera navigation. We also looked at an already implemented product similar to the Wii Remote, the 3DCONNEXION's spacemouse [1]. This device is used for smooth camera navigation. With this spacemouse you could push, pull, twist or tilt the cap of the circular "mouse" to pan, zoom, and rotate the 3D view inside RobotStudio.

During the design stage of our project we were under considerable time pressure and did not have the time to gather real user stories. Instead we based our concept ideas on the information we got from the workshop with the RobotStudio developers. We used internal workshops, discussing scenarios on what and how the users would interact with our system and a brief design document was created including common RS2008 functionality (see table 4.1).

---

[1] http://www.3dconnexion.com/

Functionality	**Description**
`Object selection`	`Select a 3D object`
`Move objects`	`Move a selected 3D object`
`Attach/Detach objects`	`Attach selected object on a robot`
`Linear jogging`	`Robot Jogging`
`Joint jogging`	`Robot Jogging`
`Camera strafing`	`Move the camera up/down and sideways`
`Camera rotation`	`Rotate the camera around an object`
`Camera Zoom in/out`	`Zoom in and out on an object`
`Start simulation`	`Robot simulation of pre-recorded movements`

Table 4.1: Implementation table

We focus our research on robot jogging and camera navigation based on many reasons. We did not want to replace a mouse and keyboard as they are 2D devices made for 2D interaction. The Wii Remote is a 3D controller that sense acceleration on three axes and can thereby know its own movement direction in 3D space. This controller was designed for gaming purposes and not for GUI interaction. For example in modern 3D games, i.e. Zelda, you navigate your character in a 3D world and the coherent camera view often follows the character. For this type of camera navigation we decided to use manipulative gestures.

The Wii Remote still have the capabilities of GUI navigation using IR-light sources which provides us with an accuracy good enough for basic deictic gestures like cursor control and object selection. To be totally free from keyboard and mouse, which enhance mobility, a replacement of keyboard interaction would be needed. We choose semaphorical gestures as the solution for this problem, were different symbolic gestures, like circles, perform different commands in the system.

Before we started our implementation we made a decision to implement all object manipulation on the main remote controller and all the camera functionality on the Nunchuck extension. The reason was to make it easier for users to remember which main context they are manipulating. The Spacemouse implementation also used this hand-dependent context scheme. The Spacemouse is used in the left hand controlling the camera, while object manipulations and GUI operations is done by the normal mouse.

4.2 Implementation

RS2008 have the possibility to load addins which we used to connect the Wii remotes with. An addin to RS2008 is a .dll file that RS2008 loads at start up. Addins can access all the internal functionality, like navigating the camera, controlling 3D objects and manipulate the current user interface. All this is possible because of the .NET platform which R2008 is based upon. RS2008 developers also provided us with a great C# API documentation of all the common functions useful for our concept.

4.2.1 Parsing wiimote Data

The accelerometer and the IR-camera operates at 100 hz frequency[2]. This data could be sent to the computer, using a normal Windows Bluetooth connection, in bit streams. With a managed library called WiimoteLib, created by Brian Peak[3], the raw bit streams are translated into real state values.

The data provided by the library is; state changes, button states, IR-positions and accelerometer data. Accelerometer data is raw uncalibrated values dependent on the current acceleration state. This raw data includes both jitter from the hardware and noise from unsteady hands. We reduce some of this jitter and noise by taking 5-10 sample values and normalizing them. After we have a normalized value we can calculate the angle on the Wiimote compared to the earth. This angle is a value between $-\pi/2$ - $\pi/2$ and is used for manipulative gestures, i.e tilting the control may rotate an object depending on the tilt angle. Since not all angles are natural to reach with your arms, we use the sign of the angle as direction and the absolute value as the speed to rotate. The effect is that we do not need to move our hands as much.

---

[2] According to http://wiibrew.org/wiki/Main_Page

[3] http://www.brianpeek.com/blog/pages/wiimotelib.aspx

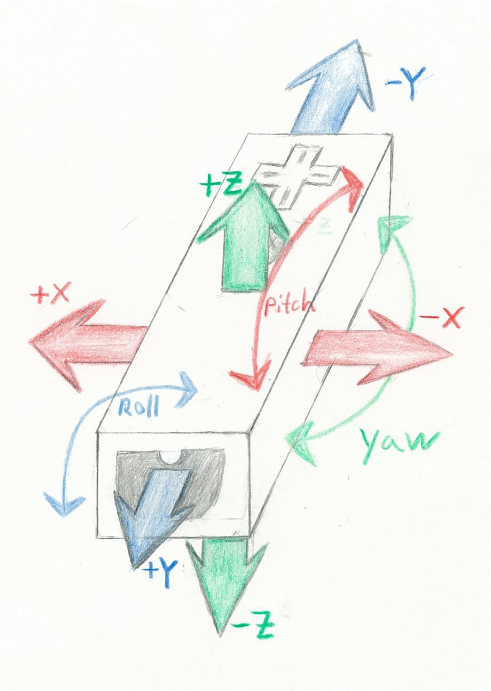


Figure 4.1: Wiimote Axes

There are three methods of how you can tilt a wiimote, pan, roll and pitch. A pan movement is a rotation around its vertical Y axis, roll is the rotation around its horisontial Y axis and pitch is a the rotation around its horisontial X axis, (see figure 4.1). Roll and pitch movements could be calclutated using a Wii remote in its normal position. To calculate roll and pitch we use the formula:

$$P = Atan(Z/Y) + \pi \tag{4.1}$$

where Z and Y are values based on how much gravity force affected on each wiimote axel.

We calculate the roll values by the formula

$$R = Atan(Z/X) + \pi \tag{4.2}$$

4.2.2 Our Architectural Design

We structured our code so it would be possible to extend functionality without rebuilding the foundation (see figure 4.2). The overall base class is called WiimoteControll and handles the startup/shutdown. It also has a WiimoteCollection class that contains a list of all the connected wiimotes.

For every wiimote their is one profile. We use profiles to be able to use the wiimotes simultaneously (multiplayer). We also use profiles to be able to limit or extend functionality for different purposes. The two implemented profiles was RobotController and SwordFight. The playerBaseProfile contains the basic timers and controll implementations, like mouse simulation. All the Mechanism functions, like calculating new robot positions, was wrapped into our own Robot class. In addition to the mechanism functions it also contains state information, for example what part on the robot that is currently selected.

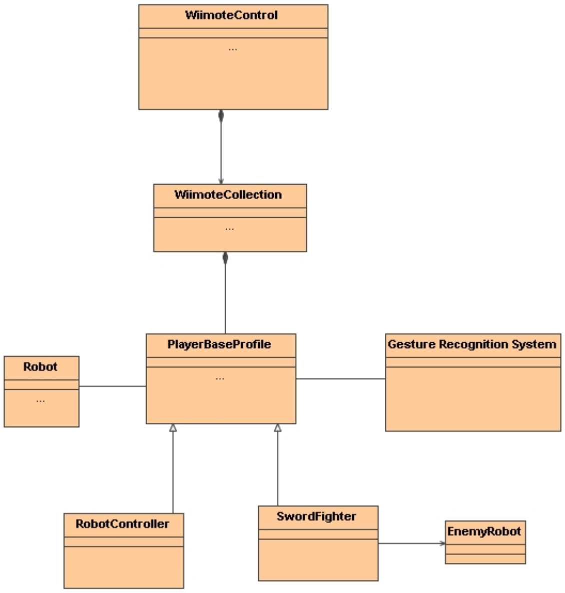


Figure 4.2: UML overview

4.2.3 Mouse Simulation

The wiimote IR-camera has a horizontal field-of-view of 41 and vertically 31 degrees. For mouse simulation we need the camera to capture one IR-source inside its field-of-view to place the mouse cursor somewhere on the screen. The IR-source's position is translated to a proper position.

Some initial problems occurred when we tried to point close to the edge of the screen, or tried to point at a small spot. Also, the offset from the middle of the screen and the middle of the sensor bar creates a small artefact which users need to adjust their aim for. Using more than one IR-light can cause the pointer to jump when the IR-camera looses the currently tracked light source and starts tracking a new one.

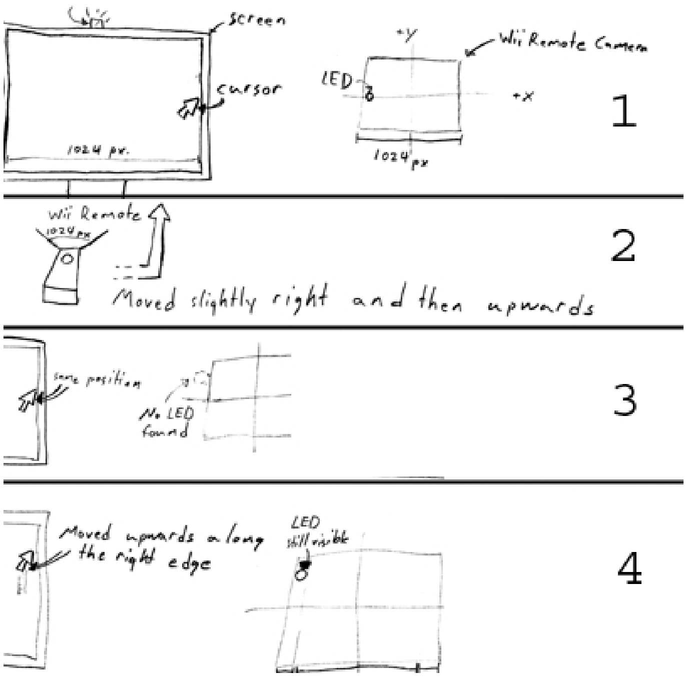


Figure 4.3: Sticky Edges

When moving the cursor close to the edge of the screen, we will have the light source in the IR-camera's corner. If we go outside the IR-camera's field-of-view we will lose track of where we are pointing, which will result in the cursor not moving. This gives us a "sticky-edges" feel when trying to manipulate toolbars along the border of the screen (see 1,2,3 on figure

4.3).

By using a small portion of the cameras outer field as a safety margin we can avoid the sticky edges. This is done when transforming the camera's IR-source coordinates to screen coordinates. First, we translate the camera center coordinate to our screen coordinate's origo. Then we scale it to the size of our screen, but add some percentage of the screen's width or height accordingly to the screen axis. We then translate the scaled camera center point to the screens center point. The increased scale will create the margin around our screen. Values outside the range of the screen will be converted to values just inside the range, giving a more natural response to the user's movement at the cost of higher sensitivity due to the reduction of the camera's active field-of-view (see 4 in figure 4.3).

Pointing with the IR-camera manually will cause the cursor to shake noticeably on-screen. We would like to filter out these motions without affecting responsibility or precision too much. We developed a simple smoothing method, which calculates the delta vector between the current cursor point and the next. The delta vector represents the relative movement needed to move to the next cursor point. The code snippet implemented is as follows:

```
double smoothFactor =
Math.Sqrt(deltaX * deltaX + deltaY * deltaY)*2
/ Math.Sqrt(screenSize.Width * screenSize.Width
+ screenSize.Height * screenSize.Height);

if (smoothFactor < 1.0)
{
deltaX *= smoothFactor;
deltaY *= smoothFactor;
}
```

where smoothFactor ranges from 0.0 and upwards, but all values above 1.0 is treated as 1.0 so we do not move past the next cursor point. The nominator represents the length of the delta vector multiplied by a weight established by empirical testing where a larger weight gives faster response at the cost of less smoothing. The denominator was set to represent the largest move that can be done in one frame, i.e. the diagonal of the screen. This can be empirically tested, but we chose to base it on the user's screen size since this will vary from system to system and just use a weight to tweak the response. This hides all noticeable shakes and gives better responsibility at higher speed.

To avoid the offset we could make a screen which emits IR-light in the same manner as the sensor bar, or we could simply try to calibrate away the offset in the software as Nintendo does. Calibrating the offset would, however decrease the vertical range of the Wii Remote's camera. Since the calibration is software based, we will still loose track of the sensor bar when we pitch the Wii Remote too much. If this calibration is needed, we found that it might be better to put the IR sensor a bit too high than too low since the users' wrist can bend upwards more than downwards.

4.2.4 Camera Navigation

One benefit with the Wii Remote is the possibility to extend its functionality with other tools like the nunchuk extension which Nintendo uses as a navigation tool in many 3D-games. We made use of the analogue stick to pane, orbit and zoom the camera in RS2008.

We choose to pane in a horizontal plane by using the Nunchuk's analogue stick which has a horizontal layout and can be tilted in a circular plane. This way we can move around like standing on a floor but without turning around, which is a good starting point while trying to avoid disorientation.

By pressing the "C" button on the Nunchuk, we enabled orbit horizontally and vertically with the same analogue stick as with our pane functionality. Our orbit was bound to the current selected object, which would cause an instant jump from any view to a view with the selected object centered in our view. It should be suggested that this kind of sudden transformation of the camera view is to be animated [19].

The same "C" button as orbit also activated manipulative gesturing by reading the pitch of the nunchuk which was mapped to a zoom in / zoom out functionality. To make the zoom more user friendly we set the current pitch when the button was pressed as the neutral zoom value. To increase the zoom the user needed to pitch downwards and vice versa. This way the user could be unaware of their current pitch and just press the button instead of first keeping its hand in the right pitch followed by the press of a button. To make the most of the accelerometer, we wanted to use the roll of the nunchuk as a manipulative tool as well, but discarded this idea due to the high risk of unwanted roll input when pitching. We therefore choose to use the roll as semaphoric gesture input as a constraint. Two semaphores were used; Lock View and Recall View to make navigation safer as well as faster.

Since we could not separate rotations and linear movements of the accelerometers Lock View was implemented as a static semaphore where the user rotated the nunchuk and tapped the "C" button. The semaphore mapped to Recall View symbolized a sideway shake. Since we had both the shake and zoom on the same accelerometer, we had to complement the nunchuk accelerometer with a cool-down function that disabled accelerometer parsing for a short period of time after a shake to prevent accidental double-shakes or a shake followed by an unwanted zoom.

4.2.5 Jogging a Robot

As described in the background, there are three ways of freehand jog a robot in RS2008. We decided to implement joint jog and linear jog.

4.2.5.1 Jointwise Jogging

Jog joint is the easiest function to implement as you only need to change the position of one joint at a time. The tricky part was to actually select one of the joints. Our solution to this is the same as the built in RS2008 joint jog. First a part of the robot needs to be selected and then you are able to move it by rotating it around its joint. In figure 4.2.5.1, you can see that the third link is selected and rolled around joint nr 2. A robot part can only be

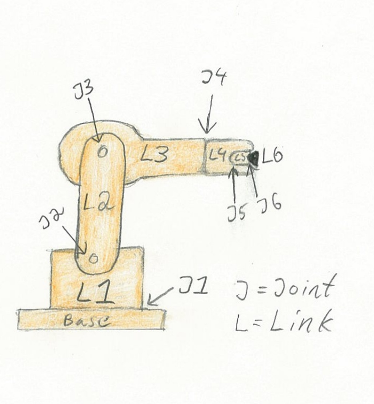
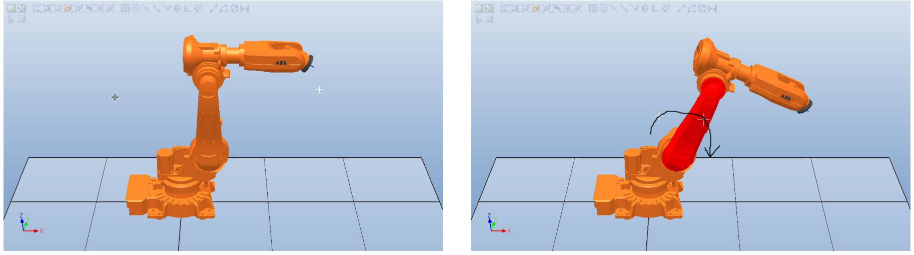


Figure 4.4: Robot Joints

moved in one direction but some links rotates around different axis. So we made link 2 and 4 rotate depending on the moving factor from roll and the rest on the pitching factor. This should create a more intuitive controlling experience since you get a greater affordance. Rolling your arm looks like you roll the robot arm and pitching the Wii remote creates a animation as shown in (figure 4.2.5.1).

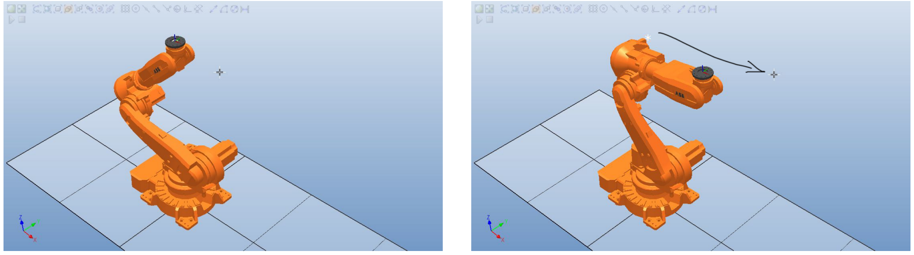


(a) Home position (b) Jog joint...

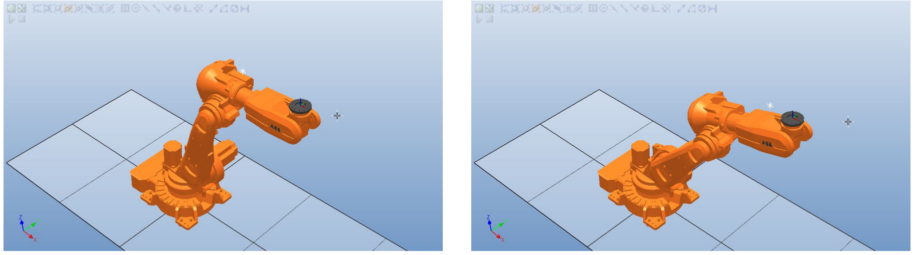
Figure 4.5: Example of Joint Jogging in RS2008

4.2.5.2 Linear Jogging

Jog linear was implemented as an option to the jog joint method (see figure 4.6). First we used accelerometer data from pitch and roll but realize that it was only a hard and intuitive way of controlling. Instead we created a 1-to-1 mapped controlling function that uses IR for more accurate data. By vertical up and down movements we translate the flange on a robot, according to a basic difference calculation between two recorded IR points. This translation make the flange move to another position and by using a Inverse Kinematic calculation, viable in RS2008, all links could be translated accordingly.



(a) start position (b) jog forward



(c) jog even more (d) end position

Figure 4.6: Example of Jog Linear in RS2008

4.2.6 Gesture Recognition

We received data from the Wii Remote at 100 Hz. The raw data structure contained acceleration data in all three dimensions and was placed in a list sorted by time. We enabled recording by saving this list to a file for later use as a reference. We could then match a recorded list with an input later on to see if the two input lists were similar.

The stored data was simplified with the least square algorithm, which we used to group ten samples together to speed up the matching process. By measuring the difference between the two lists, we could find the best matching gesture recorded earlier.

One issue we had was that the gravitation weighted the gesture data sometimes more than the actual gesture, making the level of noise quite high. Nintendo has addressed this issue with expansion hardware to filter away some of the noise, and increasing the precision.

The semaphorical gestures that was recorded for the demo was: A circle for starting the simulation, a catching and a releasing gesture for attaching/detaching a selected object on the robot, a shaking gesture with the nunchuk to reset the view and a M-like gesture for opening a menu. The shaking gesture was an importent way to help new users to get out of trouble when they where lost in the 3D space.

Chapter 5

Results

5.1 Implemented Functionality

We implemented the following functions inside RobotStudio 2008:

- Moving windows cursor.
- Camera rotation, up/down and sideways.
- Camera Zoom.
- Camera strafe.
- Save camera view.
- Get back to last saved meny by doing a shake.
- Selecting objects/menu items/robot links
- Joint jogging
- Linear Jogging (1-to-1 mapping)
- Fast link-traversing,up/down
- Save different robot poses
- Gesture recognition system
- Make the robot move between saved positions by doing a circle gesture
- Attach a selected object on the robot, doing a catch gesture

- Detach object, doing a release gesture.
- Opponent robot can explode and a score based upon how good your strike it will be calculated.
- GUI items for loading the addin, connecting/disconnecting wiimotes and swiching between profiles.

5.2 Demonstrating our Design

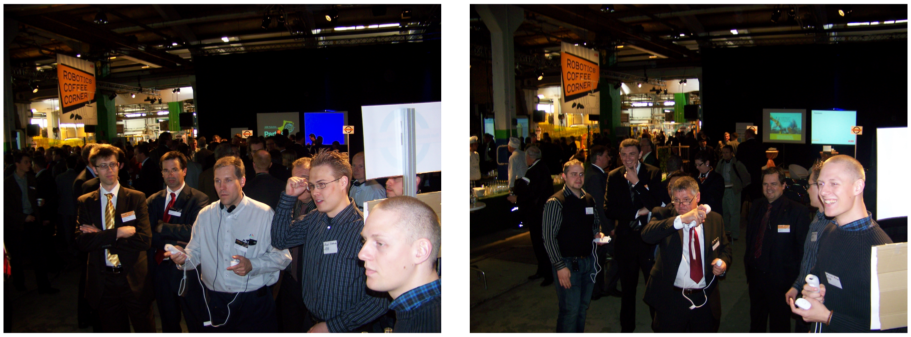
In May 2008 we demonstrated our prototype at the Robotics Partner seminar 2008 (see figure 5.1). It was a great opportunity for us to show and ask ABB customers and robot engineers about our prototype. To get more attention we created a small game in RS2008; a sword fighting ABB robot that could smash down one static opponent robot. A score was calculated if the swing was hard enough together with a nice exploding animation of the opponent robot.

Figure 5.1: Robotics Partner Seminar 2008

We do not know the exact number of visiting users but we had an official five hours demo time and we always had someone at our station watching, trying or discussing our prototype. We introduced our solution by showing a few glimpses of our prototype followed by an introduction to our work by explaining how the Wii Remote works. After our introduction we let them try the demo hands-on. The features we displayed were camera navigation, robot jogging, gesture recognition and sword fight.

5.3 Observations

Since most of the users who came to our station was looking for more information, we had to spend a lot of time showing and describing this new technology. Some were very eager to try out and discuss our solution, so we did our best to improvise questions relevant for user testing.



(a) Deictic gestures (Pointing around) (b) Semaphorical gestures

Figure 5.2: Users trying our prototype

5.3.1 Mouse Navigation

The user was standing 3 to 6 meters from the screen and navigated in RobotStudio by pointing and selecting on objects (see figure 5.2(a)). Everyone was able to navigate and select big objects like the Robot. Some were able to select different parts on the robot. Only a few managed to select the small GUI icons.

5.3.2 Camera Navigation

The joystick on the Nunchuck extension was the first thing people learned to use and it looked intuitive for almost everyone. Zoom and rotate was tricky to explain and for users to find but as soon as they did it was easy to use.

5.3.3 Jog Robot

Our most discussed functionality during our user test was the jogging functionality. We saw that joint jog was easy to comprehend. Everyone succeeded to click on a part and jog it by tilting and rolling the control. Some had problems knowing in what direction to tilt but after a couple of tries they managed to get it right. You could also see that it was easier to point the Wii Remote up than down when tilting.

Linear jogging was the most discussed function as it seemed to be the most interesting feature for many of the customers. Our implementation of linear jogging wasn't fully stable as we could not reduce all the noise, lag and the "get out of IR range"-issues. Anyhow there was a huge demand on testing it. Our 1-to-1 mapped solution was surprisingly un-intuitive. Users tried to move around the robot by moving the Wiimote in different arbitrary directions. Since we only let the flange move forward/backward or up/down according to if the users made a corresponding move, it makes it impossible to do "free" flange moves. This was a bit confusing for many of the users who actually wanted to move the robot like if they were the robot themselves. We also noticed that using all axes at the same time when jogging was hard. When users wanted to move forward they got a slight up/down movement as well.

5.3.4 Gesture Recognition

We only had gestures using two hands at this demo. It was enough to watch how intutive it was using gestures for basic commandings (see figure 5.2(b)). We noticed a difference in age. Young people mastered gesture recognition better than old. We also noticed that the Nunchuck extension cord was often in the way when users tried to comprehend some of the gestures.

5.4 Overall Feedback

The Wii Remote as an alternative to the Flex Pendant could be applicable in industrial painting. A couple of painting representatives requested another way of robot programming. Their suggestion was based on using the Wii Remote as a real world coordinate recorder. They wanted to move the Wii remote around the objects and create robot paths based on the location of the Wii Remote. This idea was a big contribution for further work in this area.

Using a Wii Remote as a robot programming tool would open the possibilities for cooperative work. Robot programmers could work together and non-engineers are able to concept their own stuff. Instead of having one engineer sitting with RobotStudio, creating a robot cage while the customer hangs over his shoulder, they could both work together in one common environment.

Many thought it was interesting with a more intuitive tool. We heard that other ABB departments had also thought of the Wii remote as a possible tool for robot programming. They were gladly amazed that we had an actual prototype to show.

Experienced Flex Pendant users were thrilled to see our prototype but were concern about the accuracy. They thought it would be hard to do the real accurate movements. They also suggested a plane-locking feature for the linear jogging function. It would be a useful ability to lock the robot to certain planes to stabilize the shaky hands and to get high precision on robot paths. This feature is common in normal robot programming, according to the robot programmers we spoke to.

Chapter 6

Discussion

Just by the working demo itself, we have shown a possible solution to the problems defined in the introduction. With two or more Wii Remotes it's possible to be multi-interacting with a system. This solves the problem of communication. A Wii remote is a small device, compared to a Flex Pendant, and uses a cordless Bluetooth connection. This solves the problem of flexibility. Many users saw the possibility in the Wii Remote as a tool for robot jogging. It was intuitive and easy to learn for most users. This shows that Wii Remote interaction, in some sense, could solve the problem of expertise.

Altough there are many things that could be improved for a further system and one of the biggest issues was the direct 1-to-1 mapping feature for jogging the robot. Enhanced algorithms would of course minimize the issues but would not solve the fact that human hands are shaky and therefore accurate movements would be hard to achieve. A solution for this problem could be a discrete step function, using the cross-arrow button for example. This would give users the option to jog the robot with predefined length steps.

The biggest issue we found with the concept of using a Wiimote Remote as an industrial tool is the common use of 2D GUI. A redesign of the GUI in RobotStudio would be necessary for efficient workflow with the Wii Remotes. Sliding bars and navigating in hierarchical menus were tricky even with a smooth cursor algorithm.

Our gesture recognition part shows the possibility to command and interact with RS2008 without using the keyboard. Using more advanced math and algorithms like HMM would result in a better recognition. The recognition could be as good as 98% [11]. This would result in higher accuracy on semaphorical gestures and thereby make it more intuitive since exact movements would not be necessary. A full scale gesture-language could then be implemented in RobotStudio to replace much of the button clicking interaction.

Other improvements that should be done in an improved addin:

- Make sure that users get relevant feedback as soon as the IR-Camera is out of reach.
- Improved feedback system for the gesture recognition.
- Improved linear jogging algorithms.
- Ability to lock linear jogging to a plane.

Since most of the users at the ABB partner seminar were more interested in a real robot prototype, further research could be focused on real robot jogging rather than improving the RobotStudio addin.

Another use of the Wii Remote, inside the domains of industrial applications, could be the suggested point marketing. Instead of jogging the robot from position to position, you could mark points on real world objects by moving the controller around them. This idea is not only in the heads of industrial painters but also in robot researchers. The ability to just walk around and create virtual robot paths is something that was so interesting that we base a whole further work chapter on it, our redesign rationale. This type of system is close to the Global positioning system available for navigation purpose. Accurate coordinate tracking tools could already be found in the industry. Laser tracking tools is already common for Robot calibration [15].

Chapter 7

Redesign Rationale

7.1 Background

The need for robots in smaller companies increases, and this changes the requirements of robot applications. Smaller companies have a more dynamic demand, changing production cycles more often while having less room for hiring dedicated robot programmers. The complexity of instructing the robot must decrease while not increasing peripheral costs.

The included possibility of having a mobile tool like the wiimote is that we can cooperate much easier. Setting up a workflow for a production line might be more intuitive when you are able to synchronize the instruction sets between robots by working many persons in parallel. Although the need is not one to be seen today, the possibility could have positive effects on robot programming. The workflow of two programmers instructing one robot each could be compared to choreographing a pair dance.

7.2 Concurrent Technology

The common desktop workstation today consists of mouse, keyboard, screen and sometimes speakers. Working with a horizontal touch screen as Microsoft's Surface [1], or something as daily as our mobile phone, will impose different work models in which the role of the Wii Remote will need to adapt

---

[1] http://www.microsoft.com/surface/press.html

to. Some of the more suited systems could include Augmented Reality feedback that can manipulate real world objects as instructed by the users Wii Remote.

7.3 Suggested Wii Remote Changes

The cord lessens the reach and hampers mobility. If two controls are needed, one composite solution has been suggested in the form of the Darwin control which is two controls put together.

The IR pointer has good precision but will restrict the user to point at a certain area. If we could triangulate the device's position without restricting the user's arm movement to the direction of the light source, we could get a much more intuitive interaction.

By changing the pitch, and maybe even the yaw, of the mounted camera in comparison to the Wii Remote we could get a better offset mechanism with fewer flaws. If motors are added to the camera, we can get a very large field-of-view, although with the same offset limitations as before if we want to maximise our range.

7.4 RoboBrush

A common scenario that we encountered when listening to current robot programmers is that they use robots to paint objects. Although time consumption or accuracy is not a big issue, some users expressed the need for non-programmers to be able to instruct the robot themselves and then let an operator fine-tune the robot's path. Some requirements on the tool as expressed by the users were 1-2 cm accuracy, both orientation and position tracking relative to another objects surface as well as velocity sensing to mimic the speed of a brush stroke. In this section we will purpose an redesign rationale that we call "RoboBrush".

7.4.1 Application Specification

The RoboBrush application consists of a spatial augmented reality projector and the Wii Remote. The projector will display different textures directly upon real objects, enabling us to get the immersive feel of actually colouring something. The Wii Remote can sense its position and rotation relative to the surface of the projected surfaces. A rudimentary way of doing this would be for the Wii Remote to parse the coordinates of the texture projection as seen by the camera of the Wii Remote. A prototype can simplify this by letting the projector put marks on the texture, thus telling the Wii Remote camera what coordinate of the object it is filming.

7.5 Scenario Based Design

To extract concrete design suggestions from our abstract ideas we create three scenarios. These scenarios will be the fundation of our redesign.

7.5.1 Personas

Norman, 35, is a well-known cartoon sketch artist hired by ”Cars R Busses” which specializes in manufactoring vans and mini-vans. Norman knows most of todays technologies needed to sketch and color cartoon comics, but knows nothing when it comes to robotics.

Robert, 52, has been programing and operating robots for over 25 years, and is a part of the Cars R Busses robotics crew. He normally instructs the robot how to paint, and often reminds the designers what is possible and what is not.

Will, 27, has been working for the research department at Cars R Busses for almost three years. He has worked with developing the RoboBrush application and will now manage the first commersial project together with Robert and Norman. Will has a very good insight in how RoboBrush works.

7.5.2 Scenario: Learning the Tool

Will, Robert and Norman are in a studio where Will has prepared a simple example by hanging up a front door of their new Van ”Revolver” in the air. Will picks up his Wii Remote, walks up to the door of the Revolver and says to Robert and Norman: ”Before we colour, by programming the robot with a Flex Pendant. Everything was supposed to be very exact from the get-go, and not very artistic. This way of doing things will be more like ’shooting from the hip'." "Now let's say we want a nice blue colour", Will do a circle gesture to choose the marking tool, and the entire door is coloured grey. He presses a button and follows the outline of the door, colouring everything but some edges of the door's outer side white. Will does a straight line with both hands to "smooth" the selection, making the last missing surface to turn white as well.

"Cool!", says Norman, "But you better have a good palette of colours, or am I supposed to gesture every nouance there is?" Will marks a square on a table just besides the Revolver door and then writes "C O L" in it. The Marked surface turns into a palette, where Will now can pick any shade of blue he likes. He picks a turquoise shade and whips the Wii Remote towards the door. The white-marked surface now turns into the chosen colour as Will smirks "like so..?" to Norman.

Robert, however, instantly points out that this is not something that couldn't be done before with normal Robot programming. "Ok, but what if we want to be a bit more creative?" asks Will. " Let's throw some more paint on there to make it more interesting!"Will makes a circle on the table, next to the colour palette and writes "B U C K E T" in it. He then drag-and-drops an orange colour from the palette to the bucket, turning it into a colouring bucket with orange colour in it. "Since I'm not the artist of us three, I will let you get to work then, Norm" says Will.

Norman Grabs the Wii Remote a bit hesitantly and asks what he is supposed to do. "Just dip the remote in the bucket" says Will. Norman moves over to the flat, orange circle and puts the top of the remote on the circle. "Oh, now you've gone and soaked the entire control in paint", Will smiles and points at a projected label that reads "Soak: 92%". " The Remote you are holding has an internal camera in the front. If you get close enough to the circle representing the bucket, you get colour on your brush. If you move the remote even closer it will get soaked even more. Now if you think it has too much paint on it, you can just shake some off and watch the label if the soak-level is good enough. Then try and throw some paint on the car as I did earlier."

Norman shakes the Wii Remote, and as he does the soak-percentage moves down until Norman stops shaking, leaving the soak-level at 52%. "Okay, so now I just whip it on the car?", Norman asks Will. Will nods. Norman makes a quick flick with his wrist and watches as the door gets covered with a variety of spots. He shakes the soak-level down to 5% and throws some color onto the edges of the car, but this time the spots pulses back and forth.

Will tells Norman that since the robot which they are using cannot produce such small spots, those elements needs to be adjusted. ”You can just leave those there, and let Robert take care of it since he knows the limitations of the robot.”

”But as you said before, Will, Norman is the designer, not you nor I.”, asks Robert and continues, ”Shouldn’t the design be all done when the work comes to my desk?”

Will goes to a nearby desk and takes out a second Wii Remote, presses the connect-button, and hands it over to Robert. He tells Robert to draw a horizontal rectangle, and write ”R O B O T” in it, then do a capture gesture to say ”I am the robot”. As Robert does so, he changes the mode of the Wii Remote so he can instruct a robot’s path preferably in RobotStudio or any other virtualization tool that enables playback.

7.5.3 Scenario: Collaborative Design

Norman has been put to work and is feeling comfortable with the RoboBrush application. Robert however is not participating much which worries Will. To convince Robert that RoboBrush is viable, he needs to make Norman and Robert cooperate a bit more.

”Hey, Robert! Can’t you help Norman out? I bet you could prepare some colour for him.” ”So what colour do you need Norman?”, asks Robert. ”Just blend something that matches with the blue and orange”, Norman responds.

After watching how to mix colours, Robert confidently gets to work. He first creates three buckets, puts colour in two of them from the palette, and then soaks the remote with the first colour, flicks it down into the empty bucket, making the empty bucket a ”colour mix bucket” which can mix different colours depending on the soak and colour contributed to the colour mix.

After adding the second colour and then a bit more of the first colour again to give the colour just the right touch, Robert tells Norman that he is done. ”OK, then you just need to point at each other and then Robert throws his current tool to you, Norman, which then catches it to mount it as your current tool.”, Will instructs them both. ”But what if you want to access both tools? Shouldn’t I be able to just lay this brush down?”, Norman asks

Will. "Well, then you just make another rectangle object and leave the brush there, but since Robert is up to no good, you two can just exchange items for now. If you just do a throw gesture instead of catching Roberts, you will do a trade instead of a catch."

"Hey, Norman, point back, will you?" Robert encourages Norman. Robert and Norman get a pleasant xylophonic chime from their remotes. Will nods supportingly as to signal "go on" to them both? Robert "throws" at which Norman responds back with the same gesture. Another chime can be heard, and a positive Norman tries out his new brush as he expresses: "I see the possibilities here. Even a guy like Robert can actually give me a hand.", Norman continues with laugh in his voice, "Although, even I can see that Roberts choice of pink isn't the best match for this car."

7.5.4 Scenario: Adjusting Robot Instructions

Robert wants to know how his daily routine will change when using the RoboBrush application. Although it looks good for a specific task like designing the paintjobs directly on the cars, the question is if Robrt can be satisfied when programming the robot.

"So each layer that Norman has saved can be accessed and adjusted on a lower level, and this is where we still need you Robert." "Oh, so I'm still needed? Well, some good news then I guess...", Robert frowns. "Even though we could simulate the painting instructions, we cannot safely optimize paths and still guarantee a good paintjob. That is up to you.", Will responds. "Do you feel like trying to paint that first blue layer then, Robert?" "Nah, I'm better off if you show me first", Robert discards Will's request. "Hmm, well I haven't done this a lot but I guess we can start out by painting the outer lines..." As Will sets up waypoints with his Wii Remote, Robert notices that they cannot actually see anything happen. "Some kind of feedback could be usable here" Robert murmurs. "That is still an issue for us, since we wanted to move away from the normal screen, we can't simply use every flat surface around as a virtual user interface. For system feedback like bad gestures or other warnings, we still feel that a normal flat-screen does its job better than our newer inventions. Feedback like where I'm placing waypoints needs to be in 3D to be seen intuitively, which most often means in 3D as well. If you use one of those helmets on the steel bench over at the door, you can get visual feedback through the transparent face-visir."

"But how come you don't project the waypoints as with the colours and the other items?", Robert asks Will. "The main issue is that we cannot project in thin air, and we thus needs to do it virtually. The face-visir has a receiver and a transmitter that will mix the real world with the virtual, and show dots where you placed waypoints." "In 3D?", asks Robert. "Yes, In 3D!" "Since we know where the door is and we film with the Wii Remote, we can determine our relative position to the door and then ask for the absolute coordinates of the door to calculate where the robot should spray from.", says Will. "But what if I require a specific pose for my robot?" "You can either choose a joint that you want to adjust, or simply freehand jog the entire robot or a subpart of it. I'm sure this isn't anything revolutionary for you, Robert, but it's not the functionality that we offer. It is a more fluent solution to package the functionality in, so that it is easier to give you robot programmers more alternatives."

Will takes out his cell phone and starts up a camera application with a grid. " Let's say you want to adjust the waypoints we just placed. I can just hook up this camera to my remote with an extension adapter cord or by Bluetooth. Since I am a fan of wireless, let's do that." Will connect his mobile with the Wii Remote, telling the mobile application what waypoints are present. Will goes closer to the door, showing a smaller part of it on the screen. With his right hand he marks a waypoint by looking on his cell phone, held in his left hand.

"By using the remote I can snap the waypoint to another grid, so by moving closer or zooming in the picture, the grid size represents a smaller area. This gives us higher accuracy. Also, if you feel a bit shaky on your hand, you can just take a still-picture and then choose a grid to move to." Robert, listening with one ear or less, knocks on the top of a red helmet lying on the steel table. "So why did you mention these helmets?"

Well, as we are working with new technology, some tools we have used will not work that well today, but holds promises for tomorrow's products. The helmet does basically the same thing as the cell phone but better." "But if it's so much better, why not use it instead of the cell phone." "We want to integrate the normal person into new environments. By using familiar tools, we try to give new users a familiar experience. When head-mounted hardware gets integrated into everyday merchandise, we will surely use it as well."

7.6 Redesign Descisions

The information described in our RoboBrush scenarios was extracted into a number of decisions. In this part we will talk about the decisions we made. A structured tree-map overview of these decisions can be seen in A. Each following subsection will describe a usability issue and promote a specific solution as well as point out other solutions as well. The main purpose of these details is to provide a template for designing a specific application which means that we leave application specific details out and focus on the general traits. If these traits fit a specific application, then the general solution should fit as well.

7.6.1 Aesthetic and Minimalistic Design

We have strived towards a ubiquitous system that is integrated into the work environment. We have thus chosen to interpret ubiquitous as "aesthetic and minimalistic" in this stage of the design although this will need more details as the design matures. To specify more, we want to base our system on intuitively, mobility, co-operability, cost and availability. The theme of these cornerstones is to minimalize constraints on the user and by making it ubiquitous also presenting it in an aesthetic way.

The system hardware consists of projectors and PC's, which are very mainstream and thus available. The PC will be used to parse data, while the projector will be part of the projected augmented reality solution. This hardware should not be in the center of attention so the user can avoid direct interaction with them.

The user hardware is Wii Remotes and cell phones, which are relative cheap in the eyes of a small company and are very mobile. The Wii Remote, which is the core of the system, can be mapped to our natural body language to create a more aesthetic feel.

Another choice could be to replace the Wii Remote and projector with some large touch screen. The benefits would be that we make the system more ubiquitous by using fingers as an input source instead of a hardware control like the Wii Remote. The downside is that this system would be bound to the output source, i.e. the screen. If the screen is not small enough to carry, it will not be mobile and thus transfers the center of attention from the user to the workstation.

7.6.2 Visibility of System Status

The Wii Remote only gives us efficient input, and we need to return proper output. The Wii Remote can give haptic feedback as well as audible. The limitations with haptic feedback are the small amount of feedback signals which we can differentiate between as well as recall. The sound feedback can be recognized instead of recalled, but might be hard to detect in certain noisy industrial environments.

Projected augmented reality can be used to incorporate a visual feedback system into our working environment. The benefits would be that we do not work against a screen, but against a real environment. The negative part is that although we free ourselves from screens, we still need to keep the visuals bound to a surface in the proximity of a projector. This will decrease the mobility of our system. For specialized tasks where we need to be more mobile but do not need many different layers of feedback, we can use a cell phone as visual feedback. This will introduce one more product for the user to handle at the added benefits of a portable feedback system with visual, audible and haptic possibilities.

7.6.3 Match Between System and the Real World

A benefit of using a projected augmented reality based system is the small step from the real world. We are able to project digital information directly onto physical objects and interact with them in a similar way as we normally would. Many handheld physical tools can be simulated with the Wii Remote in its original form or by adding different shells to enable different look and feel to it.

7.6.4 User Control and Freedom

The user will be able to manipulate fully in three dimensions, giving us the opportunity to manipulate objects by varying our distance, which is harder to do precisely when the 3D view is presented on a 2D screen. Giving the user one more dimensions to interact in will increase the complexity when only extending the behaviour to another dimension, for example, by extending a mouse cursor to point in 3D. A better way might be to still point in 2D and using the third axis to create a click with variable strength. The freedom is limited mostly by real world limitations as lifting something heavy or moving to view something from a different angle, but is also limited by the projector's displayed area.

7.6.5 Consistency and Standards

Our solution depends heavily on gesturing and direct manipulation. We would need to create a naming convention for gestures and avoid ambiguous use of the hauptic and audible feedback. Direct manipulation of objects is of importance when interacting with real objects. A delay here would break the bind between the real world and the system making the system less intuitive.

7.6.6 Recognition Rather than Recall

By giving the user contextual feedback of what gestures are possible to do at a certain state, the user can learn new gestures by looking at the alternatives and thus recalling the gesture. If we have lots of different states, we might have a hard time finding the right state/context to go to. If a context has too many gestures, it will be hard to display them all in an efficient manner. A restriction to show the most default actions is suggested.

Haptic feedback that uses morse-like codes will need the user to both recognize and change focus from the task to its hand holding the Wii Remote, creating a disturbance in the workflow. A simple way of avoiding this is to limit the haptic feedback to only be activated for a certain exception as when the Wii Remote goes out of some input range.

7.6.7 Help Users Recognize, Diagnose and Recover from Errors

A most basic feedback should show what gesture the system registered. This will however not lead to a user friendly system on its own but simply help the user to recognize an error. By finding certain traits like very high acceleration, we can tell the user to try and go more slowly or in some other way tell the user why the system is not responding correctly. It will always be hard for the system to validate the input on its own. How can the system tell if a result was faulty or valid? This can be left to the error prevention to take care of. Those gestures that produce erroneous input can be post-edited into the right gesture. This kind of post editing puts restraints on a flexible rollback function that can replace a gesture token in the middle of a gesture string just as a word editor would let you edit any letter and not just the last.

7.6.8 Flexibility and Efficiency of Use

The Wii Remote can be used by different actors. This trait combined with the removal of the one-man workstation makes it a flexible tool in computer supported cooperative work by letting all actors use the same interface. Gesturing as input will not increase the speed of the input, but can make the workflow from concept idea to product more fluent and efficient by keeping tasks on a high abstraction level that is similar to our natural way of expressing ourselves. The efficiency lies not in the speed of the input but the natural flow from thought to input command.

7.6.9 Help and Documentation

By giving all gestures clear names, we can reference to the gestures in a structured help document. This can be viewed as a tree structure to provide the contextual overview as well as a detailed demonstration of how the gesture is performed well enough to be recognized by the system.

7.6.10 Error Prevention

To be able to differentiate valid input from invalid, we need the user to tell the system if the gesture is faulty. This can be done with a button press, but will be annoying if done on too many gestures. Limiting the validation is necessary and can be done by, for example, only asking for critical gestures. Giving unsafe gestures a higher threshold for recognition is also a solution, but will just lessen the errors and not remove them. Too high threshold will make the system non-responsive and annoying. Limiting the error prevention to non-static semaphores and unsafe actions like a wrist-flick downwards to exit an application should be a good starting point. The less critical gestures like creating an object should be taken care of by the error recovery.

Chapter 8

Conclusions

In our work we have seen that the Wii Remote could be used to interact with industrial applications without the need of mouse and keyboard. It could be a possible solution to the problem of communication, flexibility and expertise. We have shown this with our addin that enabled Wii Remote interaction inside RobotStudio. By letting ABB Robotic's customers evaluate this prototype we have received a good amount of positive feedback as well as suggestions on future work. The three main issues are accuracy, precision work and freehand jogging. This could be solved with a deeper research on jogging algorithms and redesigning the prototype. Other improvements include creating an intuitive feedback system, using better gesture recognition algorithms and a change of the GUI of RobotStudio.

With our prototype we observed that a Wii Remote was not optimal in a mouse and keyboard environment. Our Redesign Rationale chapter gives an example of a system that would be more suited. Our scenarios show a description of how the system should work and is then broken down to more concrete traits based on our overall research, user feedback and experience. These traits focus on ubiquitous design that will minimize the restraints of the workflow from idea to input. Mobility and flexibility is of importance to encourage a free thinking environment that we envision. Computer supported cooperative work and augmented reality has most methods to enable this kind of system. This founding survey, on a possible blend between Wii Remote interaction and augmented reality, could provide the industrial industry with a new way of Human Computer Interaction.

Chapter 9

Aknowledgement

A big inspiration for our work has been Johhny Chung Lee[9], who has put the Wii Remote to use in different ways including head tracking and finger gesturing. We would like to thank Bertil Thorvaldsson for seeing the possibilities in using a Wii Remote inside RobotStudio and also for inviting us to Robotic Partner seminar 2008. We would also like to thank for all the support and ideas we got from the developers of RobotStudio. Last but not least, we would like to thank Isak Savo, Rikard Lindel, Baran Cürüklü, Fredrik Alfredsson and Martin Olausson, for all their help and support.

References

[1] D. Benyon, P. Turner, and S. Turner. *Designing Interactive Systems: People, Activities, Contexts, Technologies.* Addison Wesley, March 2005.

[2] O. Bimber and R. Raskar. Modern approaches to augmented reality. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 1, New York, NY, USA, 2005. ACM.

[3] J. Carrol. Five reasons for scenario-based design. volume Track3, pages 11 pp.–, 1999.

[4] G. Fitzmaurice, J. Matejka, I. Mordatch, A. Khan, and G. Kurtenbach. Safe 3d navigation. In *I3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 7–15, New York, NY, USA, 2008. ACM.

[5] C. Guo and E. Sharlin. Exploring the use of tangible user interfaces for human-robot interaction: a comparative study. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 121–130, New York, NY, USA, 2008. ACM.

[6] C. C. G. M. P. S. Hewett, Baecker and Verplank. Acm sigchi curricula for human-computer interaction. http://sigchi.org/cdg/cdg2.html 2009-08-28.

[7] M. Karam and m. c. schraefel. A taxonomy of gestures in human computer interactions. http://eprints.ecs.soton.ac.uk/11149/ 2008-05-28.

[8] H.-J. Lee, H. Kim, G. Gupta, and A. Mazalek. Wiiarts: creating collaborative art experience with wiiremote interaction. In *TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 33–36, New York, NY, USA, 2008. ACM.

[9] J. Lee. Webpage: Johnny lee's wiimote project. http://johnnylee.net/ 2008-05-28.

[10] J. Löwgren. Interaction design, research practices and design research on the digital materials, 2007.

[11] J. Mäntyjärvi, J. Kela, P. Korpipää, and S. Kallio. Enabling fast and effortless customisation in accelerometer based gesture interaction. In *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 25–31, New York, NY, USA, 2004. ACM.

[12] S. Mitra and T. Acharya. Gesture recognition: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(3):311–324, May 2007.

[13] S. Okazaki, T. Tanaka, S. Kaneko, and A. Matsushita. Vision based environment recognition for mobile robot in irregular ground. pages 107–111, Oct. 2007.

[14] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.

[15] o. j. Rickard Lindhé. Case study, abb ger livslång noggrannhet, 2005.

[16] T. Schlömer, B. Poppinga, N. Henze, and S. Boll. Gesture recognition with a wii controller. In *TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 11–14, New York, NY, USA, 2008. ACM.

[17] T. Shiratori and J. K. Hodgins. Accelerometer-based user interfaces for the control of a physically simulated character. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, pages 1–9, New York, NY, USA, 2008. ACM.

[18] J. Spolsky. *User Interface Design for Programmers.* Apress, June 2001.

[19] S. Sreedharan, E. S. Zurita, and B. Plimmer. 3d input for 3d worlds. In *OZCHI '07: Proceedings of the 2007 conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction: design: activities, artifacts and environments*, pages 227–230, New York, NY, USA, 2007. ACM.

[20] Webpage. 3d cad and 3d studio max. http://usa.autodesk.com 2009-05-01.

[21] Webpage. Adxl330. http://www.sparkfun.com/datasheets/Components/ADXL330_0.pdf 2009-05-01.

[22] Webpage. Wikipedia; augmented reality. http://en.wikipedia.org/w/index.php?title=Augmented_reality&oldid=215345201 2008-05-30.

[23] Webpage. Wikipedia; nielsen's heuristics. http://en.wikipedia.org/w/index.php?title=Heuristic_evaluation&oldid=202150774 2008-05-30.

[24] Webpage. Wikipedia; ubiquitous computing. http://en.wikipedia.org/w/index.php?title=Ubiquitous_computing&oldid=210797925 2008-05-28.

Appendix A

Descision Map

A.1 Description

We used Jakob Nielsen's heuristics[1] as a base for our redesign rationale. The map starts with Nielsen's heuristics as problems noted on the map as questionmarks, which has some suggested solutions marked as lamps. A decided solution is marked as a handshake. Pro's and con's are coupled to solutions in form of a plus or minus, or a plus/minus if the effect can be either. Comments are marked by a purple notepads to clarify something. Some solutions implies effects outside the bounds of that specific sub-tree which was marked by a pro/con or a questionmark. These were sometimes linked to another sub-tree since the solution affected another heuristics, or simply ended with a questionmark when it got outside the scope of our design. Note that the map is not a complete one, but only describes the more noticable suggestions, pro's and con's.

---

[1] http://en.wikipedia.org/w/index.php?title=Heuristic_evaluation&oldid=202150774

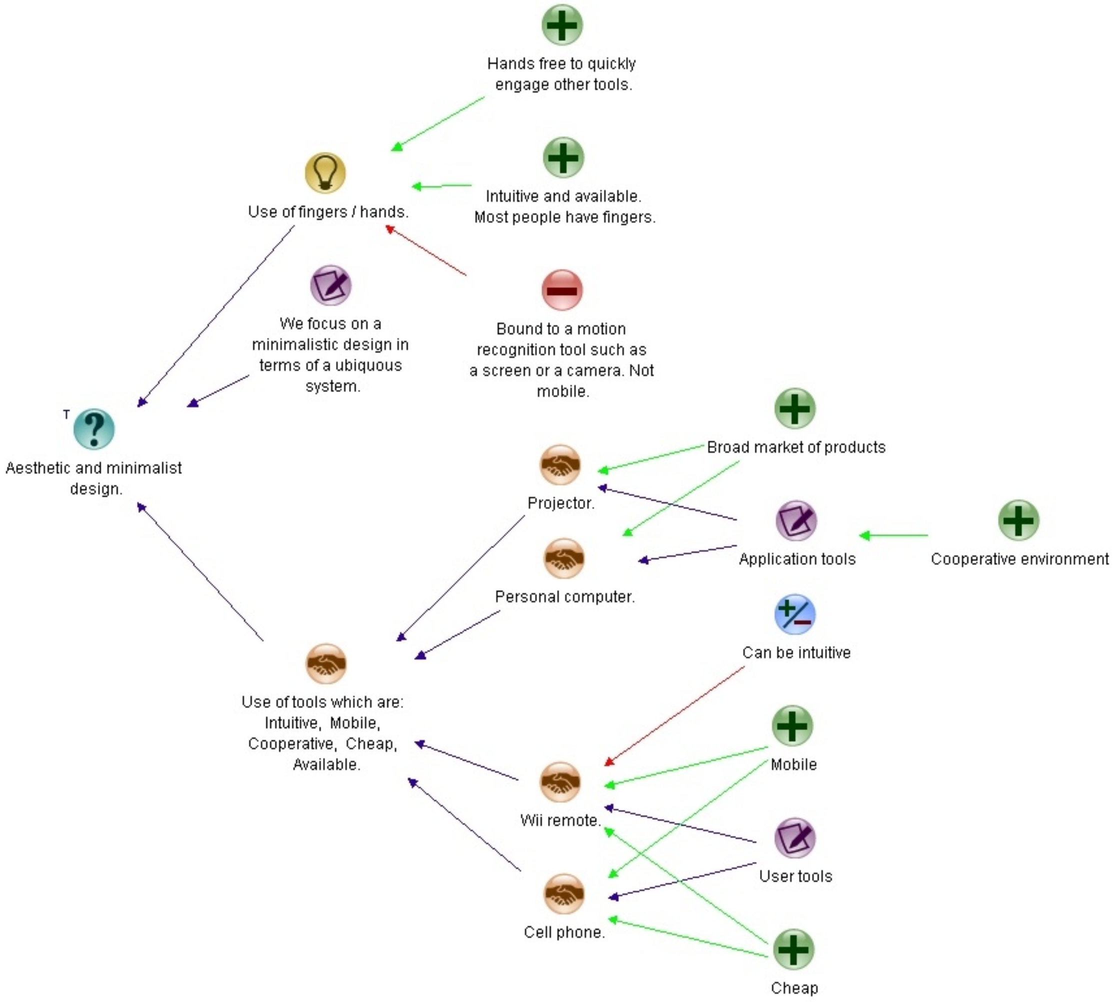

Hands free to quickly engage other tools.
Use of fingers / hands.
Intuitive and available. Most people have fingers.
We focus on a minimalistic design in terms of a ubiquous system.
Bound to a motion recognition tool such as a screen or a camera. Not mobile.
T
Aesthetic and minimalist design.
Broad market of products
Projector.
Application tools
Cooperative environment
Personal computer.
Can be intuitive
Use of tools which are: Intuitive, Mobile, Cooperative, Cheap, Available.
Mobile
Wii remote.
User tools
Cell phone.
Cheap

Figure A.1: Redesign Rationale map

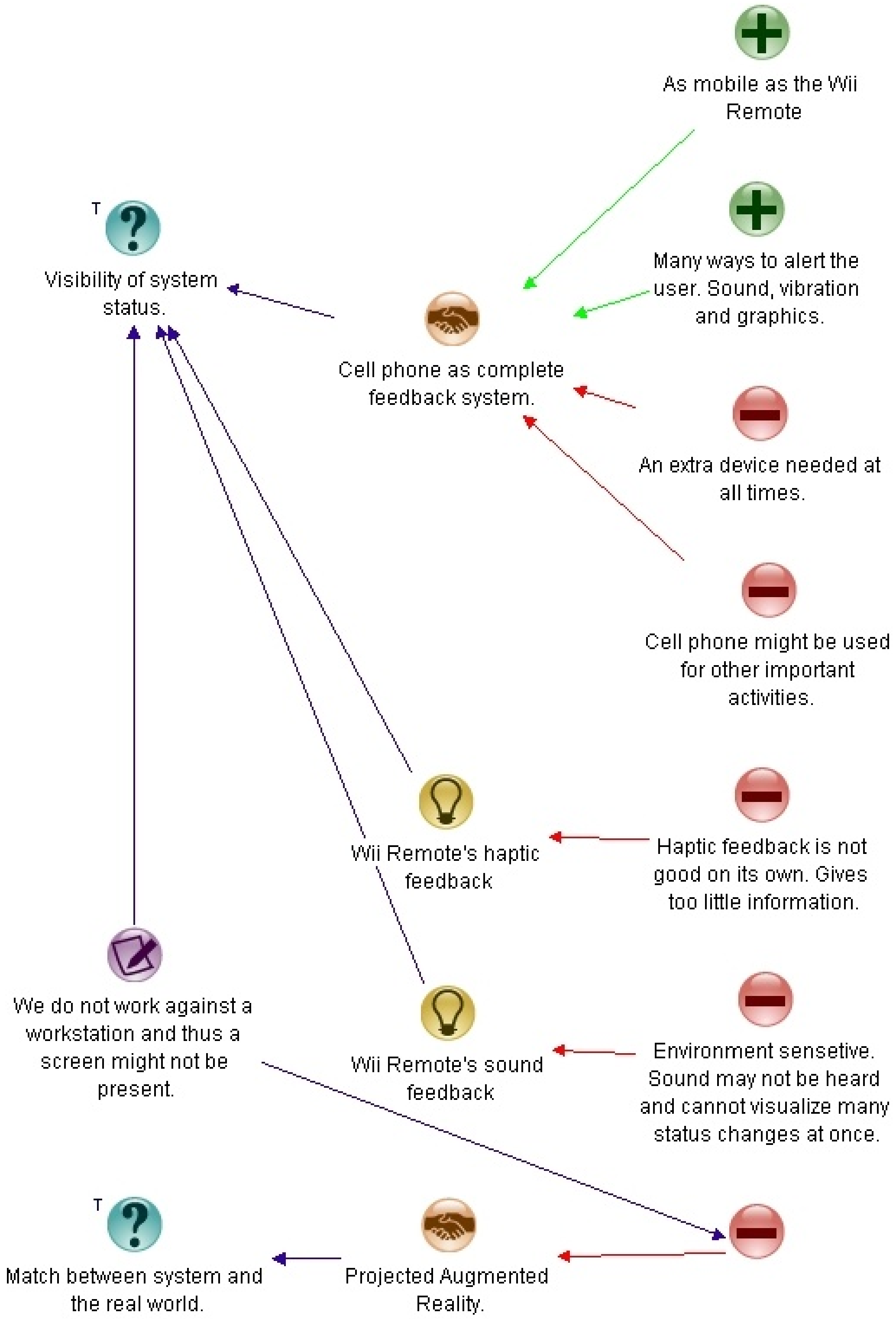

As mobile as the Wii Remote
Many ways to alert the user. Sound, vibration and graphics.
T
Visibility of system status.
Cell phone as complete feedback system.
An extra device needed at all times.
Cell phone might be used for other important activities.
Wii Remote's haptic feedback
Haptic feedback is not good on its own. Gives too little information.
We do not work against a workstation and thus a screen might not be present.
Wii Remote's sound feedback
Environment sensetive. Sound may not be heard and cannot visualize many status changes at once.
T
Match between system and the real world.
Projected Augmented Reality.

Figure A.2: Redesign Rationale map

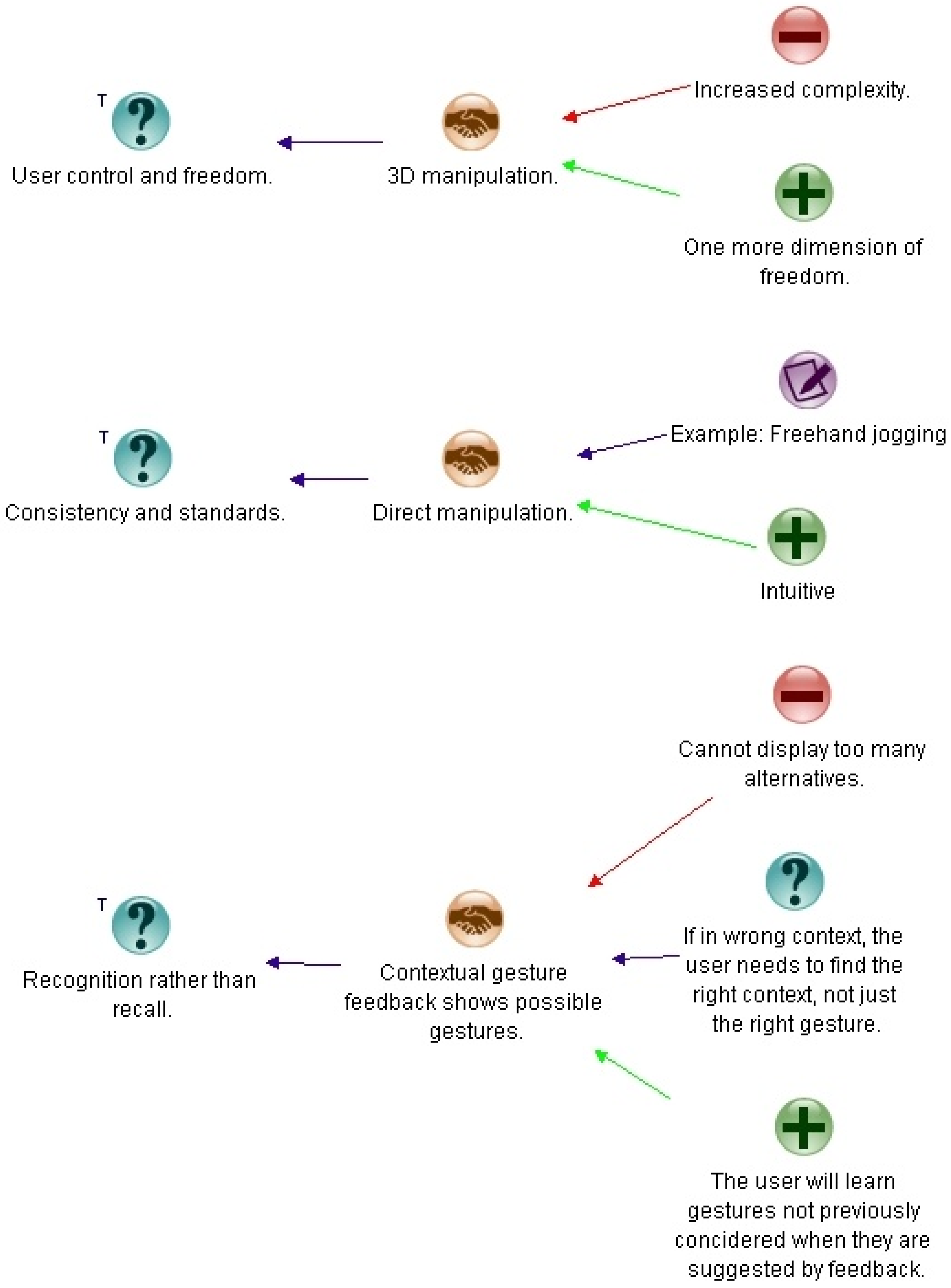

T
User control and freedom.
3D manipulation.
Increased complexity.
One more dimension of freedom.
T
Consistency and standards.
Direct manipulation.
Example: Freehand jogging
Intuitive
Cannot display too many alternatives.
T
Recognition rather than recall.
Contextual gesture feedback shows possible gestures.
If in wrong context, the user needs to find the right context, not just the right gesture.
The user will learn gestures not previously concidered when they are suggested by feedback.

Figure A.3: Redesign Rationale map

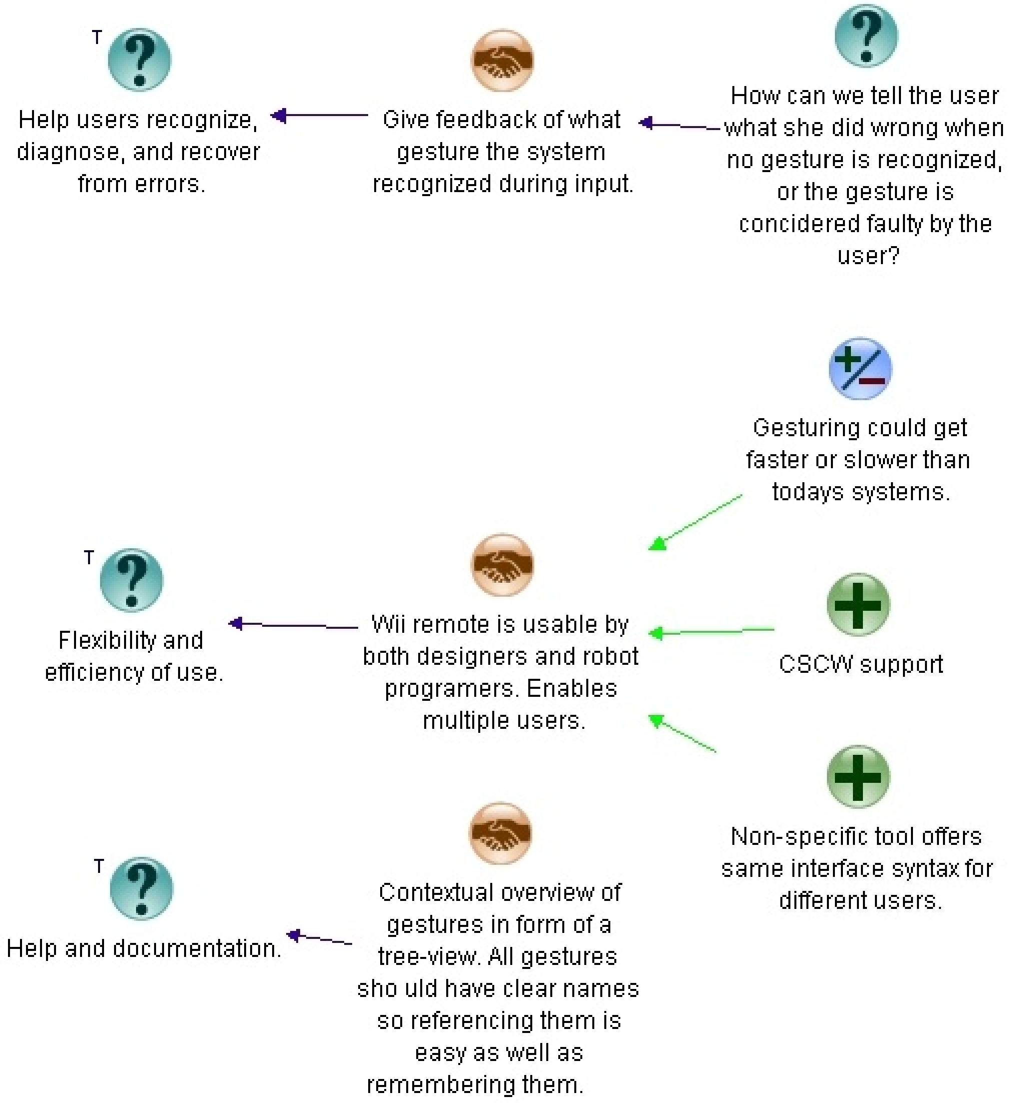

T
Help users recognize, diagnose, and recover from errors.
Give feedback of what gesture the system recognized during input.
How can we tell the user what she did wrong when no gesture is recognized, or the gesture is concidered faulty by the user?
Gesturing could get faster or slower than todays systems.
T
Flexibility and efficiency of use.
Wii remote is usable by both designers and robot programers. Enables multiple users.
CSCW support
Non-specific tool offers same interface syntax for different users.
T
Help and documentation.
Contextual overview of gestures in form of a tree-view. All gestures sho uld have clear names so referencing them is easy as well as remembering them.

Figure A.4: Redesign Rationale map

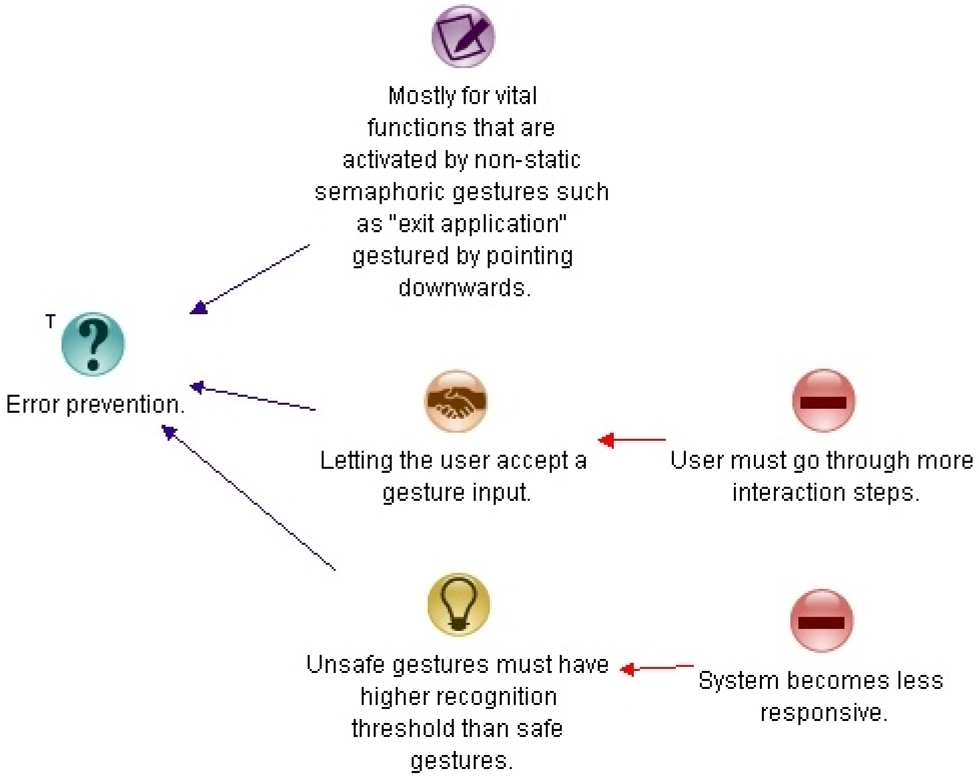

T
Error prevention.
Mostly for vital functions that are activated by non-static semaphoric gestures such as "exit application" gestured by pointing downwards.
Letting the user accept a gesture input.
User must go through more interaction steps.
Unsafe gestures must have higher recognition threshold than safe gestures.
System becomes less responsive.

Figure A.5: Redesign Rationale map