University of Art and Design Helsinki UIAH
Media lab

Master's Thesis

QuiQui's Giant Bounce

Concept and Interaction Design of a Perceptually Interactive Computer Game for Children

Perttu Hämäläinen

Instructors: Raimo Lång, Heidi Tikka

Espoo
15th November 2002

Abstract

Children's increasing use of computers can have side effects, such as obesity and stress injuries, especially when the time to play computer games is taken from outdoor activities and other physical exercise. The side effects are largely caused by the way computers are used, that is, sitting still and manipulating input devices, such as a keyboard and a mouse. However, the increasing processing power of computers and the emergence of low-cost desktop cameras (webcams), often integrated with a microphone, make it possible for the computer to "see and hear" the user through real-time mathematical analysis of audio and video signals. This enables novel *perceptual* user interfaces that challenge the user to use his or her voice and whole body. In addition to being physically stimulating, such user interfaces can make computer games more immersive and compelling. They let the user interact more naturally and directly with the game world, without manipulating any technical input devices.

This thesis describes the design of QuiQui's Giant Bounce (Kukakumma Muumaassa), a children's computer game played using body movements and voice. The game runs on an IBM PC compatible computer equipped with a webcam and a microphone. The user interface is perceptual and multimodal: the user controls a little green dragon that mimics the user's body movements and breathes fire when the user shouts.

This thesis describes the QuiQui's Giant Bounce concept and a working prototype produced. The thesis focuses on interaction design, including the user interface and technical design of the prototype. These were the author's main areas of responsibility with the addition of sound design. The prototype features computer vision and hearing technology designed for child users in real-life environments, such as homes and schools.

Keywords: interaction design, children, perceptual user interfaces, computer vision

Acknowledgements

Many people have contributed to this thesis in the form of thoughts, ideas and work. First of all I would like to thank the QuiQui's Giant Bounce team, past and present, including Johanna Höysniemi, Teppo Rouvi, Laura Turkki, Taina Myöhänen and also Tiina Kristoffersson, who worked with us on the Media lab Brand Identity Workshop, Susanna Mäkinen, whose flute improvisation can be heard on the northern desert of MuuMaa, and Sofia Koski and Antti Iiskola, who lent their voices for the watering can and QuiQui. The project also owes a great deal to the French-Finnish School in Helsinki, especially the children who participated in the usability tests and Helena Manner, who took care of the practical issues and acted as a communication link between us and the children and their parents. The help from Riikka Pelo and her wonderful children Oskari and Eemeli was also invaluable. Oskari and Eemeli participated in several informal usability tests and they also appear in the game's presentation video.

I would also like to thank professors Heidi Tikka and Raimo Lång for their guidance and suggestions, as well as the Media lab teachers Samu Mielonen, Maari Fabritius, Juha Huuskonen, Peter McGrory, Maria Koskijoki and Jukka Ylitalo. Thanks also to Markus Lamminpää for proofreading. Last but definitely not least, I want to thank my fiancee Johanna Meltaus for the crucial emotional and LaTeX support.

This thesis has been supported by University of Arts and Design Helsinki UIAH, Alfred Kordelin foundation and Yleisradion 75-vuotisjuhlarahasto.

Espoo, 15th November 2002

Perttu Hämäläinen

Contents

1 Introduction

Children's increasing use of computers can have side effects, such as obesity and stress injuries, especially when the time to play computer games is taken from outdoor activities and other physical exercise. The side effects are largely caused by the way computers are used, that is, sitting still and manipulating input devices, such as a keyboard and a mouse. However, the increasing processing power of personal computers and the emergence of low-cost desktop cameras (webcams), often integrated with a microphone, make it possible for the computer to "see and hear" the user through real-time mathematical analysis of audio and video signals. This enables novel *perceptual* or *perceptive* user interfaces that challenge the user to use his or her voice and whole body. In addition to being physically stimulating, such user interfaces can make computer games more immersive and compelling. They let the user interact more naturally and directly with the game world, without manipulating any technical devices.

This thesis describes the design of QuiQui's Giant Bounce (Kukakumma Muumaassa), a children's computer game played using body movements and voice. The game runs on an IBM PC compatible computer equipped with a webcam and a microphone. The user interface is perceptual and multimodal: the user controls a little green dragon that mimics the user's body movements and breathes fire when the user shouts.

1.1 Scope of the thesis: the QuiQui's Giant Bounce concept and a working prototype

The scope of this thesis is limited to the QuiQui's Giant Bounce concept and the CD-ROM that was submitted to the Milia New Talent 2002 competition in November 2001. The CD-ROM is attached to the cover of this thesis. It includes a working prototype of the game, an offline version of the project's website and presentation videos. Note that in the Milia version of the prototype, all text has been translated from Finnish into English. Since January 2002, the project has continued as a post-graduate research project.

This thesis focuses on interaction design, including the user interface and technical design of the prototype. These were the author's main areas of responsibility with the addition of sound design. However, the sound design was not a lot of work compared to the interaction design, so it is only briefly touched in this thesis.

1.2 Overview of the project and the roles of the participants

The thesis project started after Christmas 2000 as a collaborative project between the author and Johanna Höysniemi, with scriptwriting support provided by Laura Turkki. The first two months of the year 2001 were spent designing the story, characters, interaction and game structure. These were refined during the spring through prototyping and testing. During the spring, the team also got two new members: Teppo Rouvi (graphical design) and Taina Myöhänen (producer).

The first public release of the QuiQui's Giant Bounce prototype took place in June 2001. The interaction and game design was practically complete at that time, but the technical design had some problems. The technology was redesigned and tested and the prototype was considered to be complete in fall 2001, when it was submitted to the mindTrek 2001 and Milia New Talent 2002 competitions. Apparently the prototype had no major flaws, since it won the Pikku Kakkonen category (best multimedia for children) in the mindTrek competition. It was also one of the 14 winners of the Milia New Talent competition.

The author was responsible for the sound design, technical design and programming. The concept and interaction design was done collaboratively by the author and Johanna Höysniemi. The author also experimented with different technologies and user interface prototypes. The prototypes were refined based on the results of testing and interviews done with children, conducted by Johanna Höysniemi. Höysniemi was also responsible for the visual design principles and the design of the project's website. Teppo Rouvi produced all the graphics for the prototype. The game and character design was done collaboratively by the whole group. Laura Turkki wrote the script that the prototype is based on.

1.3 Current status of the project

The authors are currently working on a series of game prototypes similar to the flying game described in this thesis, but exploring other themes of movement and also some new areas of research. The goals of the project are still fundamentally the same, with the addition of the author's and Johanna Höysniemi's doctoral degrees. The results of the project, including scientific papers and downloadable software, are published on the project's homepage *http://kukakumma.net.*

1.4 Document structure

This thesis is divided into two main chapters. Chapter 2 describes the QuiQui's Giant Bounce concept. Chapter 3 describes the interaction and technical design in more detail. Note that although the text was written by the author, the ideas in Sections 2.1-2.6 and 2.8 were developed collaboratively by the author and Johanna Höysniemi or the whole team. Also considering the other sections, it must be acknowledged that in team work, people contribute to each other's thinking constantly. Without the discussions with the other members of the team, the author's view of the project and all the related concepts would no doubt be less insightful.

2 The QuiQui's Giant Bounce concept and prototype

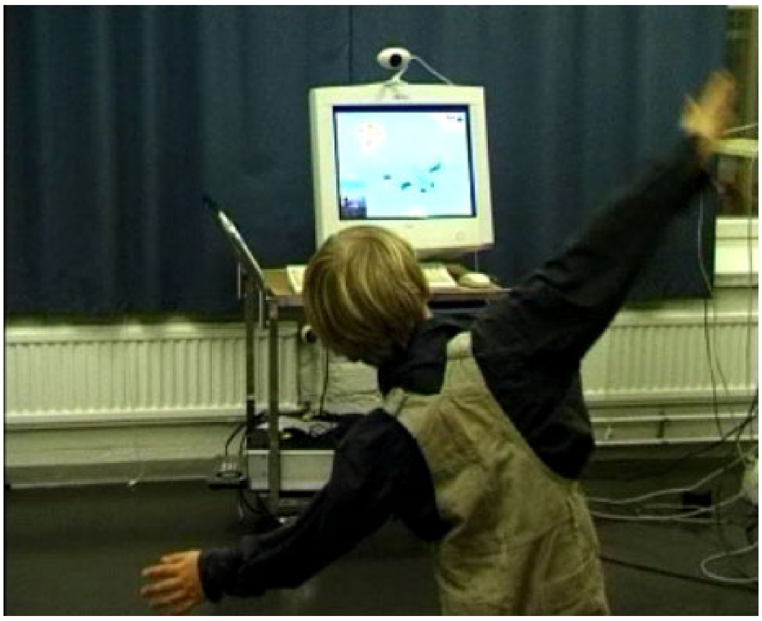

Figure 2.1: 5-year-old Oskari playing the game

2.1 A perceptually interactive game for children

QuiQui's Giant Bounce is a computer game for 4 to 9-year-old children that is controlled through movement and voice. The game does away with keyboards and traditional game controllers and uses a web camera (webcam) and a microphone to "see and hear" the player. The typical setup of the game is that the webcam is positioned on top of the monitor, as shown in Figure 2.1. The user moves in front of the monitor, facing it. The game has a series of game tasks connected by a storyline. Each task has its own theme of movement, such as flying, jumping or swimming.

The main goal of the game is to offer a physically challenging alternative to traditional computer games. QuiQui's Giant Bounce activates children to use their bodies and supports the development of their physical abilities, such as coordination skills, spatial recognition and balance.

QuiQui's Giant Bounce is an example of perceptual and multimodal interaction design. The field is quite young and it is only recently that scientific conferences devoted to perceptual interaction have been organized, such as the Workshop on Perceptual User Interfaces (see *http://www.cs.ucsb.edu/ conferences/PUI/index.html*). Perceptual interaction means that the computer perceives the user in the same way as a human would, using electronic or otherwise technically implemented senses, such as vision or hearing. The field is closely related to ubiquitous computing, in the sense that the technology is unobtrusive. The user does not need to operate any equipment, such as a mouse and a keyboard.

Perceptual user interfaces are interesting when designing computer games, since they enable the user to naturally act out the role of the main character. The natural user interface can deepen the user's emotional commitment to the game and make the game more compelling and exciting. Properly designed perceptual user interfaces can also add physical exercise to the game. This is a benefit compared to traditional computer games, the overuse of which may have negative effects on children's physical health (see e.g. Subrahmanyam [43]).

Technically speaking, this thesis deals with *computer vision* and *computer hearing*. In practice this means that audio and video signals are analyzed mathematically to extract information about the user's actions. Note that the term computer hearing is used to emphasize that it refers to a simulated sense, although terms like machine listening and computer audition are more common in the scientific literature.

2.2 The prototype: a flying game

The QuiQui's Giant bounce prototype produced in this thesis consists of the game's main menu and one game task, a flying game. The prototype works on an IBM PC compatible computer equipped with the following hardware and software:

- Microsoft Windows 98/2000/ME/XP operating system
- Microsoft DirectX 8 or higher
- Macromedia Flash 5 player
- Windows compatible soundcard

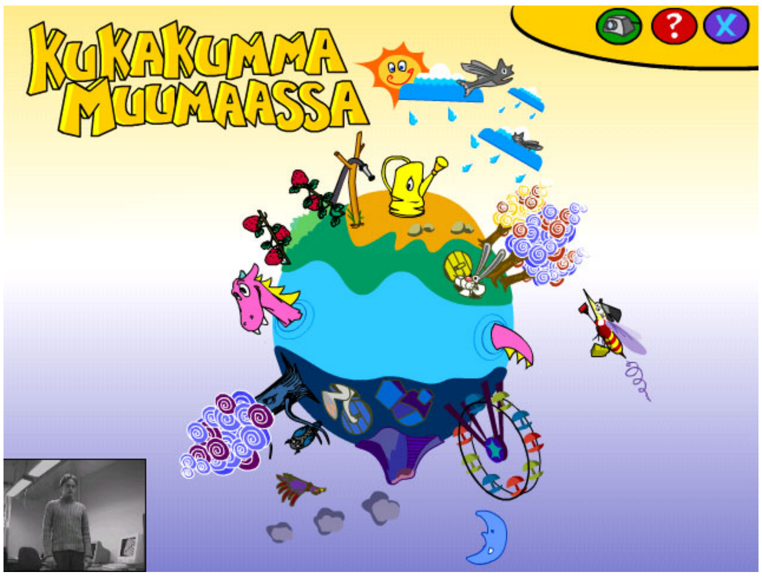

Figure 2.2: The main menu of the prototype

- Windows compatible webcam
- Microphone (optional)

The prototype is provided on the attached CD-ROM as an automatic installer package. It can also be downloaded from the project's website *www.kukakumma.net*. The website also contains instructions and other material, such as the background story, pictures and videos.

The game starts with the theme song and the menu shown in Figure 2.2. In the menu, the user can start the flying game by clicking on the yellow watering can. Other options are exiting to the ending credits by clicking the x-button, viewing the help panel by clicking the question mark button and adjusting Windows camera settings by clicking the camera button. Note that throughout the game, the bottom left corner of the game shows a webcam view so that the webcam can be placed correctly. The upper body and hands of the user should be visible at all times to make the game function correctly.

In the flying game, the player controls the game's main character QuiQui, a little green dragon. The game starts with instructions spoken by the watering can. The starting view is shown in Figure 2.3.

In the game, QuiQui has to help the yellow watering can to water the desert. This is accomplished using a pair of leaves to fly through the clouds to make them rain. A screenshot of the game in action is shown in Figure 2.4. The game ends when all the eight clouds have rained. In the end, QuiQui flies down with a loop and the watering can congratulates him, showing the time spent. After that, the game returns to the main menu.

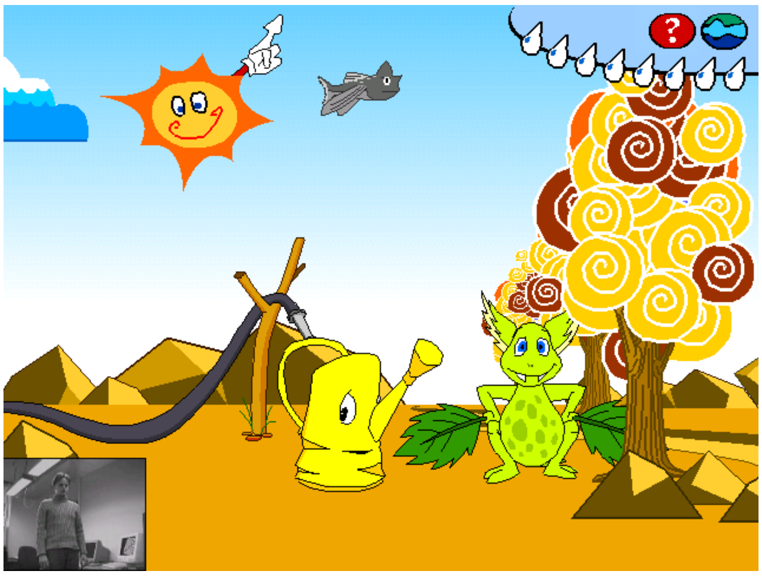

Figure 2.3: Starting the flying game

The user controls QuiQui by shouting, waving his or her hands and bending his or her body sideways, as shown in Figures 2.4 and 2.5. QuiQui moves his hands correspondingly and rotates in the direction the body is bent. Shouting makes QuiQui breath fire and sparkles.

The game has the following interactive objects, shown in Figures 2.3 and 2.4:

- The clouds. In the beginning, the clouds are filled with water. When QuiQui flies through a cloud, it rains. The surface of the water goes down and raindrops fall from the cloud. When raining, the cloud also makes a musical note. When empty, the cloud is completely white.
- The flying fish. The flying fish fly horizontally through the view every now and then, jumping and splashing in the clouds that are still full of water. If QuiQui hits a fish, he falls down a bit. QuiQui can scare the fish away by breathing fire. If a fish gets hit by a flame, it flees in the direction it was coming from.
- The sun. The sun points towards the nearest cloud full of water.
- The raindrop symbols in the top right corner. Each cloud that has rained causes a raindrop symbol to turn blue in the top right corner.
- The help and exit buttons in the top right corner. The help button displays a help panel and the exit button returns to the main menu.

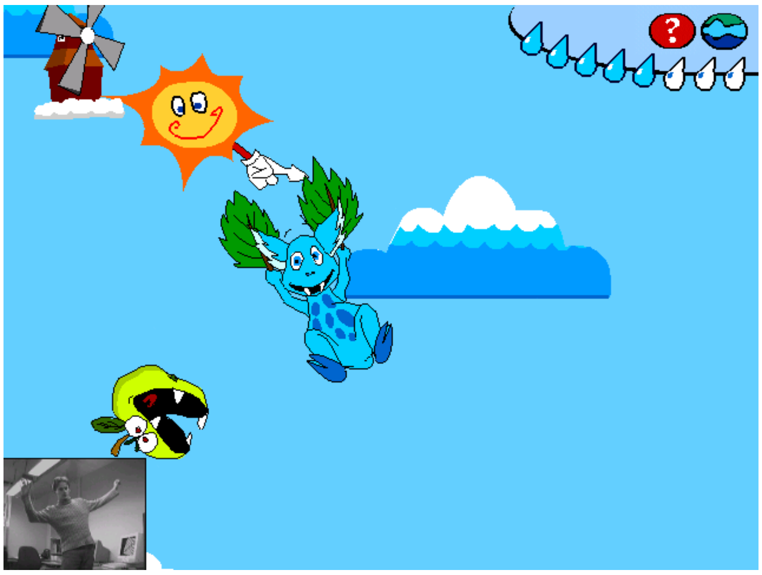

Figure 2.4: Flying through the clouds

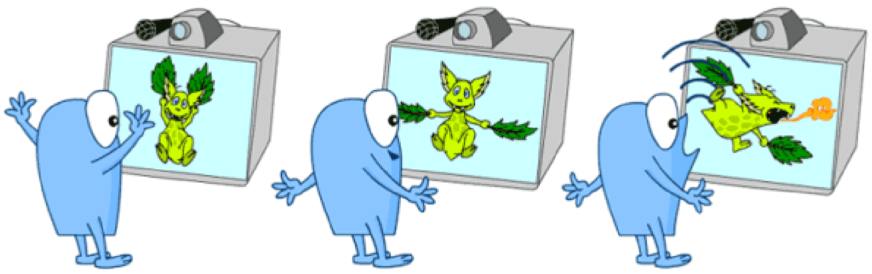

Figure 2.5: The interaction model: the avatar mimics the user's actions

- The apples. There are three apples that bounce on the clouds or on the ground. The apples are colored blue, green and pink. If QuiQui hits an apple, he eats it and his color is changed correspondingly.
- The windmill. The windmill adds challenge to the game. When QuiQui gets near enough, the windmill's wings rotate and try to blow QuiQui away.

2.3 Design goals: usability and immersion

There are quite many scientific projects dealing with perceptual interaction, such as the Kidsroom and Virtual Aerobics Teacher installations developed at the MIT Media Lab [38]. There have also been some efforts in using perceptual or physical interaction technology in commercial products, such as the camera based installations for amusement parks by Vivid Group [1], the IntelPlay camera based games by Intel and Mattel [18], and the sensor carpet based dancing game Jungle Book Groove Party by UbiSoft [2]. Our goal was to make QuiQui's Giant Bounce to stand out from other products and experiments in the following ways:

- A user interface making use of the whole body and also voice. The existing perceptually interactive games only use one input modality, usually movement detected with computer vision technology. Other games used with body movements are based on physical sensors, such as a sensor carpet, that do not provide data on the whole body of the user. The technology behind QuiQui's Giant Bounce combines both computer vision and computer hearing. It does not interfere with gameplay, as the user interface does not require any devices the player should wear or use.
- High usability, regarding both the game and its installation. QuiQui's Giant Bounce is designed for and with children, based on interviews and iterative usability testing. Designing for children from the point of view of usability also causes strict requirements on the technology, as elaborated later in this thesis. Previous perceptually interactive works all require a controlled environment or some assistance from the user when configuring the system. However, this is not desirable in software designed for children. The software design should hide all the technical complexity so that the game 'works out of the box', installing and configuring automatically.
- No need for any specific hardware. QuiQui's Giant Bounce is designed to work with all Windows compatible webcams and microphones instead of a sensor carpet or specific camera models, such as the IntelPlay Me2Cam used by Intel and Mattel. This makes the game accessible for a wider audience, since it can be downloaded from the project's homepage in digital form.

2.4 Target group and environment of use

From the very beginning, we decided to design the game for as young children as possible, since the human motoric development is fastest at a young age. For information about children's motoric development, see for example the book by Ismo Karvinen [31].

To reach as wide an audience as possible, the game is targeted for the natural environments where children use computers, that is, homes, schools and daycare centers. An alternative would have been to design the game as an installation for museums or amusement parks.

However, the interaction technology sets a lower limit for the age group. When testing different versions of the prototype with children, we found out the age of five or four to be a practical lower limit for the target group. Younger children may have problems, for example, with realizing that they should stay within a certain area since the camera's field of view is limited. Finally, we decided that our target group would be children of 4 to 9-years-old, with focus on the range 4 to 6.

2.5 Game design and narrative

When designing the game and the narrative, we tried to place QuiQui in relation to existing characters and brands, as shown in Figure 2.6. The figure has three axes: dynamics, violence and importance of narrative.

We wanted the game to be dynamic, but not as action-packed as the Pokemon television series. The game should be exciting but it should not have an excess of violence. However, it should have some edge instead of the nauseating good-heartedness of Winnie the Pooh.

The authors believe that a perceptual user interface is an excellent way of adding excitement without violence. Although the game uses cartoon graphics, it does not contain the infamous 'funny' acts of violence typical to cartoons. Also, no-one is killed in the game. The antagonists cannot kill QuiQui, only get in his way. They can, for example, make QuiQui fall down when he is trying to fly up. QuiQui cannot kill other characters, he can only scare them away or trap them for a while.

To have a compelling story was also important. Initially, we wanted to create an adventure that combines physical action with the narrative experience you get from a good book. On the narrative axis of Figure 2.6, QuiQui is closer to Disney's Beauty and the Beast than the Teletubbies. However, what we first had in mind was probably too complex. We observed in our tests that it is important to motivate children with a narrative, such as helping a sympathetic character. On the other hand, longer arcs of drama and a more elaborated story do not seem as important, which is supported by the way children themselves tell stories. A good reference to children's storytelling is the Children Are Telling project that has published children's stories on the Internet [3].

As for the structure and interactive narrative, we did not want to use a too complex or branching storyline because of the target group. Also, we had not seen any truly convincing and compelling works with a branching storyline.

The basic structure of the game is that it has a series of game tasks connected with a linear storyline. The games start with an introductory story

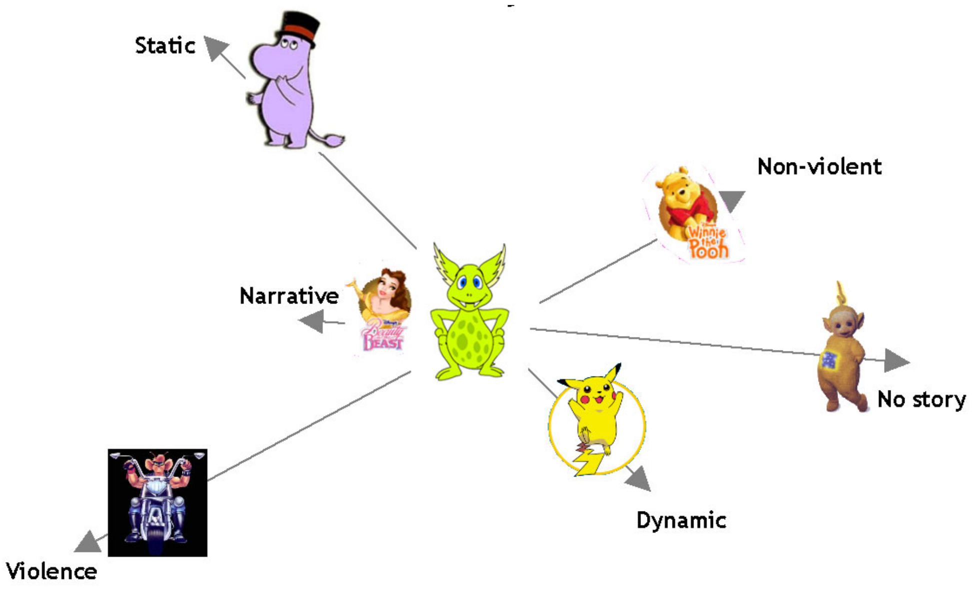

Figure 2.6: QuiQui's Giant Bounce in relation to other characters and brands

animation that explains the task, for example by introducing a character that needs help. However, the introductory animation was left out of the prototype because of lack of resources. Instead, the story can be read on the project's homepage.

The use of the animations is similar to the use of video clips between game episodes in adventure computer games, such as the Tomb Raider series [4]. The animated story conveys the drama and motivation, the game tasks themselves are small skill-and-action games. In the game tasks, the challenge is to learn to move correctly, for example to jump and run up a hill, dodging rocks that are plummeting down. The use of a straightforward storyline and focusing on the action is in agreement with the "less choice, more responsiveness" approach for physically interactive story environments proposed by Pinhanez et al. [38].

Each game task has its own theme of movement, for example jumping, flying or swimming. The first task contains only simple movements, but the following tasks require more skill and may also combine different movements.

2.5.1 Storyline

The storyline in a nutshell is that the planet Muumaa has stopped rotating so that one half of the planet lives in an everlasting day and the other half in an everlasting night. QuiQui arrives at the planet, hatching from an egg floating in the sea. He is carried to the shore by the friendly pink lizard Kalander, who also acts as a helping character throughout the game. QuiQui begins adventuring around the planet, meeting and helping various characters.

Gradually, he gets hints of what is going on. In the prototype, QuiQui has to help the watering can that is watering the desert, dried in the constant blaze of the sun. Finally, QuiQui gets into the planet's core and restarts the rotation.

The project was problematic from the point of view of scriptwriting. Scriptwriting is closely connected to interaction design and motivates the themes of movement used. However, the interaction design of a game task can be fully decided on only after the technology has been developed, prototyped and tested. We soon decided that only the rough outline of the story would stay constant, but the details would be crafted along with the prototypes of the games.

The author had little part in the scriptwriting, so the details of the story are not discussed in this thesis. However, the synopsis used when designing the flying game is included in appendix C, written by Laura Turkki.

2.5.2 Characters: QuiQui and Kalander

Each game task introduces new characters, but two characters appear throughout the game. One is naturally QuiQui and the other is Kalander, a helpful character that QuiQui meets in the beginning of the game.

QuiQui is a little, perky and adventurous green dragon. QuiQui's speciality is his rainbow-colored fiery breath that changes into bubbles under water and small clouds of ice crystals in the winter. Although he is referred to as a 'he', he was designed to be a genderless character that both boys and girls could identify with. We did not want to support choosing among multiple main characters, mainly because of limited production resources. To be more specific, QuiQui is not an ordinary dragon, since dragons usually have wings. QuiQui's wings are missing simply because being able to fly throughout the game would make it difficult to provide motivation for the other themes of movement.

Kalander is the helping character that pops up whenever QuiQui needs help. In the beginning of the story, QuiQui is floating in the sea inside an egg. He hatches from the egg when waves throw it on Kalander's head. Kalander then carries QuiQui to the shore to begin the adventure.

2.5.3 Help system

Kalander is somewhat related to the Cheshire Cat in Alice in Wonderland in that he can appear and disappear anytime and anywhere. When QuiQui needs help, Kalander's head peeks from a hole in the ground or simply from behind the edge of the screen. All instructions are spoken by Kalander or other

Figure 2.7: QuiQui

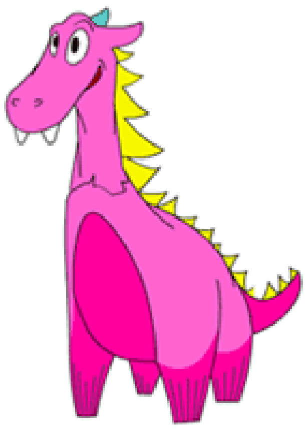

Figure 2.8: Kalander

characters QuiQui meets, whichever is more convenient considering the game context.

Game instructions are also provided using videos of playing through the game tasks. This is analogous to arcade game machines that 'play alone' when nobody is using the machine. However, in QuiQui the videos also teach how to control the game, since the camera's view of the 'model' user is shown in the bottom left corner of the screen. According to our experience, the operation of perceptual user interfaces is always easier to show by example than to explain in words.

The prototype's help functionality is reduced to a panel of text and figures with no spoken help or videos. The spoken help would have required iterative user testing and rewriting in addition to recording the speech with voice actors. The video was left out because using full screen video would have increased the download size of the prototype. The videos would be most effectively implemented by recording only the low-resolution video of the user. The game system could then replay a game using the recorded video as the control data instead of the real-time camera input. This would require some additional programming in the prototype.

2.5.4 Free navigation

The user may freely select a game task from the game's main menu. Our hypothesis is that the user remembers a game task by its theme of movement, thinking for example "It was fun to fly! I'll try that again" However, this hypothesis has yet to be validated because the prototype only features one game task.

The approach has the drawback that the game offers less challenge and reward than if the game tasks were only available after the user has completed the preceding task. However, in an environment such as a daycare center, the game has multiple users and the latter approach would need some way of identifying the users. This is problematic, since it would make it more complex to start the game. Because the target group is at least partly illiterate, the game cannot identify the users by requesting them to type in their names. One solution for identifying the users could be to use pictures taken with the webcam. This is not implemented in the prototype, but will probably be experimented with in the future. The user profiles could also enable other forms of personalization and reward. For example, the graphics of the main menu could be varied depending on how many times the user has played each game task.

2.6 Interaction and visual style

From the point of view of interaction design, our goal was to enhance immersion and the player's bond with the avatar. We wanted a transparent and intuitive user interface that would let the player to get under the avatar's skin with a minimum amount of practicing. Perceptual interaction technology seemed ideal for this, since it allows a direct mapping between the user's actions and the avatar's actions.

The basic interaction model is that the user is represented by a graphical avatar that mimics the user's actions. In the prototype this means that QuiQui flies when the user waves his or her hands. QuiQui also breathes fire and sparkles when the user shouts. This is illustrated in Figure 2.5.

The visual style is cartoonish two-dimensional animation, shown from the point of view of a third person. Although three-dimensional graphics would probably make the game more immersive, using two-dimensional graphics was more practical considering the resources available. None of the authors had extensive experience in creating three-dimensional graphics, but both Johanna Höysniemi and Teppo Rouvi were able to create two-dimensional graphics rapidly using the Macromedia Flash application. We also wanted to keep the team as small as possible, since the main focus of the project was in experimenting with the interaction.

The author had little part in designing the visuals, so they will not be discussed further in this thesis.

2.7 Sound design

2.7.1 Goals: cinematicness, naturalness

One of the first goals set for QuiQui's sound design was a certain cinematic characteristic, a semblance of film sound. Lacking a better word, the word *cinematicness* is used here.

Cinematicness seems like a prevalent goal in current computer games. The audiovisual design of computer games approaches movies in many aspects. The production budgets are soaring, the story is told using high-quality movie sequences, the quality of the three-dimensional graphics and special effects is catching up with movies, and the sound design uses several layers of effects, music and ambient sounds. Technology no more places any practical limits for game sound.

Cinematicness was set as a goal mainly because the story was to be told using animation sequences and the animations and game tasks were to be connected together as seamlessly as possible. However, cinematicness is not some strictly defined concept, but more of a mental guideline. The prototype's sound design tries to avoid traditional computer game sound elements, such as constant background music and synthetic marker sounds. The author also tried to evaluate the sound elements outside of the game context, pondering whether the elements could also be used non-interactive animation.

Another important sound design goal was a kind of naturalness and warmth, to avoid cold and synthetic sounds. This is mainly realized by the music that uses mainly real instruments or realistic synthetic sounds. In the prototype, the main instruments used are acoustic guitar, flute and soft flute-like synthetic sounds.

2.7.2 Challenges of interactive sound

Interactivity poses new challenges and restrictions for game sound design compared to film sound design. In games, the sound designer can only mix and finish parts of the soundscape, but the soundscape as a whole depends on the actions of the user. The sound design cannot be tested simply by listening, but someone has to actually play the game.

Music is probably the part of game sound most affected by interactivity. Music creates mood and ambience and it is often used to anticipate actions and mood changes before they are seen in the visuals. This is rarely possible in interactive applications, since the actions of the user cannot be predicted infallibly. The user may stay at one space or mood for a very long time so that a pre-composed theme may run out. On the other hand, the user may suddenly perform an action that calls for a rapid change of mood, that is, a transition from one musical theme to another. This requires the composer to use special methods, such as using several simultaneous layers of music or specifying all the possible transitions [5].

Interactive music is a research field of its own. Solutions have been proposed, for example, for automatical variation of themes and algorithmic composing of game music. One of the most widely used technical systems is Microsoft DirectMusic [14].

2.7.3 Interactive musical elements

In QuiQui's Giant Bounce, music is used primarily in the story animations and to convey a mood change, for example when the user moves from one place to another or if the user's actions start a dramatic series of events. In the prototype, the music is implemented simply by playing sound files, since an appropriate interactive composition tool was not found. Microsoft DirectMusic is not used, since QuiQui's Giant Bounce should be portable to other operating systems than Microsoft Windows. The prototype has three songs: the main theme, the ambient theme of the flying game and the ending credits.

The game tasks use musical ambient sounds instead of music with a clear melody and structure. The game tasks may last a long time and the user might get bored or irritated by a repeating melody. The flying game has an ambient musical theme composed of a flute improvisation accompanied by arrhythmic percussion sounds made with pieces of bricks and rocks. The soundscape is interactive in that the levels of musical ambience and the sound of wind are controlled according to QuiQui's position. The music dominates close to ground level, but higher in the air the sound of the wind grows louder and the music fades out.

An additional goal of the sound design is to develop the user's musicality when possible. The game tasks give feedback of the user's actions by musical means and use musically interactive objects. The prototype has clouds that each play a different note when QuiQui flies through them, so that the user can play melodies by flying. The notes fade away slowly so that it is also possible to produce chords and harmonies by flying fast enough. Another example could be to jump up a staircase where each step corresponds to a note on some musical scale.

2.7.4 Feedback: how do you know that you are walking?

In QuiQui's Giant Bounce, the user should be able to forget that he or she is standing with feet on the ground in front of the computer. The user should be able to imagine that he or she is for example flying in the virtual world of the game. The soundscape of the game has a strong effect on this in the form of aural feedback.

When you walk, the feeling of walking is mainly created through four senses. You see that you are moving in relation to the world. The sound of your footsteps tells you what kind of material you are walking on. Your muscle sense and sense of touch tell you that your feet are moving and touching the ground. However, the sense of touch is imprecise because of shoes and clothing.

The user interface of QuiQui's Giant Bounce tricks the muscle sense while the audiovisual content tries to create an illusion of movement. The role of sound is important, since the game can surround the user only using sound. The role of sound is even more pronounced in QuiQui's Giant Bounce than in traditional games, since the user is further away from the screen.

Sound has two roles in creating the sense of movement:

- Aural feedback supports the muscle sense, making the user feel the medium he or she is virtually moving in. In other words, the soundscape has to react to the user's movements. This was not fully realized until the sounds of QuiQui's 'wings' were added to the flying game of the prototype. The waving of the user's hands (the waving of QuiQui's hands and the big leaves) triggers swooshing sounds similar to a big bird flapping its wings. The sounds seem to make the game considerably more immersive. Sometimes the sounds even seem to make you feel the stronger air resistance of the game world. Another example could be the sound of water that should react realistically when the user swims.

- Sound conveys information about the movement of the avatar or the camera in relation to other objects. The sounds of the objects should move realistically in a three-dimensional space. Unfortunately this is quite difficult in QuiQui's Giant Bounce and it has not been implemented in the prototype. Satisfactory results were obtained simply by changing the volume of sound objects according to their distance from QuiQui.

Three dimensional sound can be produced using headphones, several loudspeakers or a pair of loudspeakers if the locations of the loudspeakers and the user's ears are known [30]. Using headphones is against the unobtrusive technology approach and only few people have connected their computers to a multichannel audio system, such as a home theater system. A pair of loudspeakers could be used, provided that the location of the user's ears could be precisely tracked using the webcam. Such systems have been studied by Gardner [24], using the LAFTER computer vision lips and face tracker [36]. However, this would require information about the camera's optics and the location of the loudspeakers. Precise tracking is also difficult with low quality webcams. The upper body and hands of the user must fit in the picture so that the user's face may be represented only by a small number of noisy pixels.

2.7.5 Advanced interactive audio

In addition to simulating the medium of movement, the data from the user's movements opens up other possibilities for sound design. The user could control the music and sound effects to create a new living and reacting dimension into the game. For example, the rhythm of the music could be linked to the rhythm of the user's movements. The synthesis of music and sound effects could also be controlled directly by motion in the same way as in the legendary Theremin [6]. The Theremin is an electric musical instrument controlled by the position of the player relative to two antennas. Its eerie sound was quite popular in old science-fiction films.

Microphone input is also an option that has not been utilized in game sound design, at least to the author's knowledge. It could be used to create spatial effects such as echo. Game characters could also use it, for example to repeat the player's words backwards or store pieces of sound and play them back to other users.

Unfortunately, not all of the presented ideas could be implemented in the prototype. New methods and applications of interactive game audio are currently one of the author's main areas of research.

2.8 Ideology: non-commercial research vs. designing a product

The author would like to conclude the concept description with some thoughts on financial issues. During the project, it was often discussed whether the game was going to be commercialized in some way. We felt that the project could really make the world better, at least in some tiny aspect, so we wanted the game to be freely available. We considered several ways of combining free distribution of the game with some earning logic, such as

- Bundling the game with appropriate hardware, such a computer or a webcam. At first, this might seem like a brilliant idea. However, bundling deals are sought after by several software companies and the profits made are small. Software may even be bundled free of charge to promote other products by the same company or more advanced versions of the same software. There is also a danger that a camera manufacturer might steal the concept, because they usually have their own development teams specialized in creating applications for their cameras. Many cameras come for example with some motion detection software, meaning that camera vendors do have at least some expertise in computer vision.
- Shareware. The usual notion of shareware is that you get to try the software for free and if you like it, you voluntarily pay a small amount to support the authors. Ideologically, it is a nice effort, but in practice the payment system rarely works, at least to the author's knowledge. However, it might be that shareware does not work simply because making the transaction requires too much effort. Not all people are willing to give their credit card numbers to small software vendors and international bank transfers are cumbersome. Shareware could be perhaps revived using mobile transactions, for example using text messages.
- Combining the game with a television show. The story animations could for example be shown weekly in a children's program, after which the game could be downloaded to a computer or a set top box from the broadcaster's website. Also, the webcam could act as a new feedback channel in interactive television shows. For example, a selected number viewers could compete against each other, live from their homes.
- Using the game as a proof-of-concept product for a technological platform. The computer vision and hearing components used in QuiQui's Giant Bounce could be licensed to other companies as a perceptual interaction platform. The companies could then develop their own products using the same technology. The QuiQui's Giant Bounce game itself would act as a demo project. However, considering the installed base

of webcams, large scale commercial product development is currently probably not very lucrative. Also, focusing on technology would not be artistically very interesting.

- Free national distribution but commercial international versions. As a children's product, QuiQui's Giant Bounce relies heavily on localized speech. The Finnish version of the game would probably be not very interesting for English children. The Finnish version could be distributed freely, but international versions would be sold through an international publishing company, such as UbiSoft. In a way, Finland would act as a huge testing laboratory so that the commercial versions would have less bugs and unfortunate surprises.
- Sponsorships. Since QuiQui has had good publicity and it has the potential to benefit children's physical health, it could probably attract sponsorships from companies. Product placement could also be exploited more in computer games. The author only knows only of one campaign where a game was given to all customers of a supermarket. The game was a rally game, where the player had to collect packages of products sold at the supermarket, driving around city streets lined with commercials. The game was probably a modified version of some old game. Once a game has been programmed, it requires relatively little effort to change the graphics and sounds.

We ended up doing the project with a minimal budget, supported by our part-time jobs and public funding. It seemed to be the most productive and free way. Also, none of us was ready to found and manage a company. As anticipated, the novelty and free availability of the game brought it a lot of positive publicity in the media. However, the publicity was only national and it seems very difficult to make the project known internationally without a proper marketing budget. The future of the project is open and much depends on the research currently pursued.

3 The interaction design of the flying game

3.1 Starting points and motivation

The motivation for this thesis was the author's artistic ambition combined with the concern for the increasing amount of time young children spend with computers and game consoles, instead of playing outside using their bodies and practicing their basic motoric skills. The author's background is in game design as well as electronics, programming and multimedia signal processing. During his MA studies at the UIAH Media lab, a goal slowly formed in the author's mind: to combine sweat and physical action with computer games.

As an example, the author wondered how to bring the action and aesthetics of movement of martial arts computer games back to the real world. The aesthetics seem to have first been exaggerated and transformed from real-world martial arts to Hong Kong action movies and adapted from them to computer games such as Tekken. The author wanted to develop systems that would make it possible for the user to experience the freedom of movement possible in the virtual worlds of computer games, governed by freely designed game physics instead of the unyielding physics of the real world.

3.1.1 Ujo Sankari: a physically interactive platform jumping experiment

The concrete starting point of QuiQui's Giant Bounce was Ujo Sankari (Shy Hero). It is the author's first perceptually interactive game experiment made for the Chrismas 2000 Demo Day of UIAH Media lab. Figure 3.1 shows a screenshot of the game. Ujo Sankari is a prototype of a physically interactive platform jumping game. The game character jumps on various graphical objects and it is controlled by jumping in front of a webcam. The name of the game stems from the brown paper bag covering the hero's head, used because the author could not draw a face for him. Fortunately, the game encouraged Johanna Höysniemi to propose that we would work together, which lead to the founding of a more widely talented team.

Johanna Höysniemi realized that Ujo Sankari was an example of a multimedia product that would not smother children's spontaneous physical energy. Instead, such products could provide physical exercise and challenge for children and support the development of their basic motoric skills. Höysniemi was searching for a topic for her final thesis, her ambition being exploring new forms of interactive narrative for children. Höysniemi also had the crucial experience of working with children, as she had been a swimming instructor for several years.

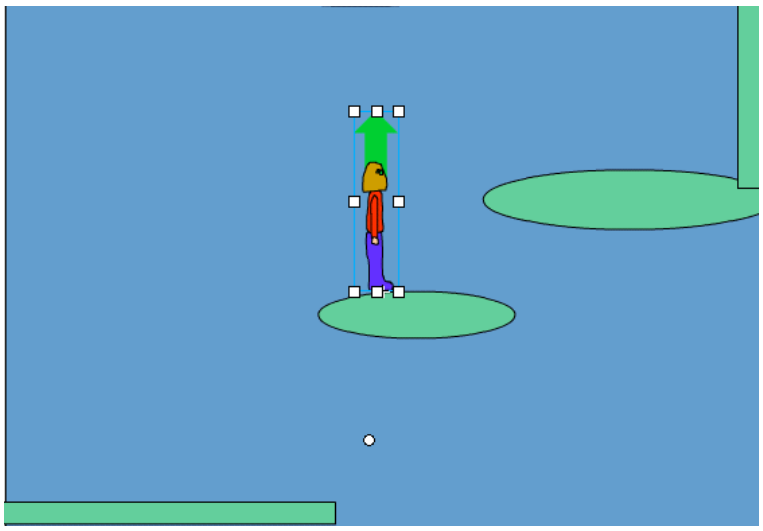

Figure 3.1: A screenshot of Ujo Sankari

3.1.2 From jumping to flying

The author had been programming and demonstrating Ujo Sankari at the home of Laura Turkki, who in a gust of inspiration wrote a series of game scripts during her Christmas holiday, including the basic idea of the flying game. The original idea was to jump on the clouds to shake the water down. Platform jumping changed into flying in the prototype when it was decided that QuiQui's Giant Bounce would contain several game tasks with different themes of movement. We had to choose one theme for the prototype and flying felt like the one that would catch the attention of users and media.

The multimodal user interface concept of flying and breathing fire was a synthesis of several ideas. The author had been playing the medieval fantasy adventure game Drakan, where the user controls a flying and fire-breathing dragon with a mouse and keyboard user interface. The game was fascinating and together with earlier experiments with voice-based user interfaces, it gave the author the idea of breathing fire by shouting. The idea could be used when the QuiQui character was born after several attempts with penguins and other creatures.

3.1.3 Computer vision and hearing as the technical approach

QuiQui's Giant Bounce is based on computer vision and hearing. The game perceives the user by mathematically analyzing the video and audio signals produced by a webcam and a microphone. Using computer vision and hearing was a practical choice, since much of the technical framework had already been implemented for Ujo Sankari. However, there are also other alternatives, such as ultrasonic tranceiver systems or gyroscopic sensors. For an overview of different technologies for tracking human motion, see e.g. Mulder [7].

Computer vision and hearing were chosen as the interaction technology because we wanted to provide a natural and unencumbered user interface that would also be robust and accessible in every home. We did not want to use any sensors attached to the user's body, which is awkward if several children want to play the game together or by taking turns. Computer vision and hearing are also ideal technologies for the interaction designer. In principle, they enable the game to sense all that the designer senses.

The equipment needed is an IBM PC compatible computer and a webcam. The microphone is optional, enabling the fire-breathing functionality. Although the game would probably be more immersive as an installation with a large, projected screen and three-dimensional multichannel audio, we wanted the game to be accessible to as large an audience as possible.

Because of the variety of webcams on the market, we did not want to optimize the game for a particular hardware setup. Supporting as many camera models as possible poses a challenge because the cameras differ a lot in terms of noise, color accuracy etc. However, it allows the game to be delivered in digital form via the project's website, with no physical gadgets packaged with it.

The computer vision was programmed in C++ in both Ujo Sankari and QuiQui's Giant Bounce. Other than that, the games combine several software technologies. More information about the software implementation can be found in Appendix A.

3.1.4 Computer vision and its problems in Ujo Sankari

Ujo Sankari is an example of using simple computer vision in an artistic installation. Although the technology works in an installation, it would be problematic in software targeted for children in real-life environments. Identifying and solving the problems was perhaps the biggest challenge in developing QuiQui's Giant Bounce. The author programmed Ujo Sankari on a whim in two nights, but developing the computer vision for QuiQui took several months.

The computer vision in Ujo Sankari is based on a technique called *background subtraction*. When the game starts, the first frame of the camera input is saved in the computer's memory. It is assumed that the frame contains only a static background image without any users, as shown in Figure 3.2a. For each subsequent frame, a *difference image* [41] is computed, shown in Figure 3.2b. The intensity of each pixel in the difference image equals the absolute value of the difference between the intensities of the frame and the background. Finally, the difference image is *thresholded*, as shown in Figure 3.2c. The output of the thresholding at each pixel is one if the intensity of the pixel is greater than a threshold value and otherwise zero.

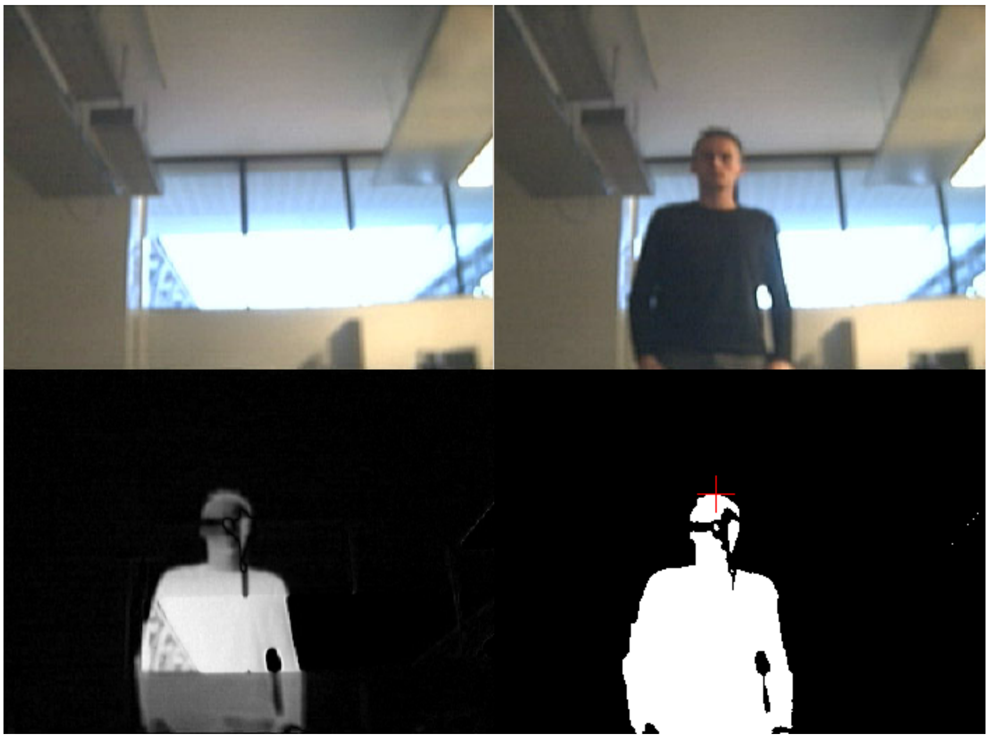

Figure 3.2: The computer vision in Ujo Sankari, from top-left to bottom-right: a) the background, b) a frame showing the user, c) the difference image, d) the thresholded difference image, with a cross marking the location of the user's head.

The information needed for Ujo Sankari can be computed from the thresholded difference image. All that is needed are the X and Y coordinates of the user's head. If it is assumed that all nonzero pixels belong to the user and the user's hands stay lower than his or her head, the coordinates of the topmost nonzero pixel can be used, marked with a cross in Figure 3.2d. When the user's head moves up, the avatar jumps at a speed equal to the head. The horizontal position of the head controls the horizontal airspeed of the avatar. Thus, the character can be controlled by jumping or running in situ.

Considering QuiQui's Giant Bounce, the following problems were identified in Ujo Sankari:

1. The use of background subtraction means that the game must obtain a clean sample of the background. When the game starts, the user must not stand in front of the camera, which is not desirable if a five-year-old child is using the game without adult supervision. A child might not understand or remember the instructions. Using background subtraction also requires that the background and lighting stay constant and the camera does not move during the game.
2. The threshold value had to be set by hand. This is too technical for the target group. All such settings should be done automatically.
3. Due to the noisy image of the webcam used, Ujo Sankari sometimes jumped spontaneously when the noise exceeded the threshold value. The game should perform image filtering or other operations to reduce noise. The analysis should also be based on larger scale features than a single pixel to reduce the game's sensitivity to noise.

3.1.5 Related work

Modelling human perception

The scientific literature reveals much work done in the field of perceptual user interfaces based on computer vision and hearing. In general, current computer vision and hearing technology is far behind human perception. Although much can be learned from testing with human and animal subjects, all stages of perception cannot yet be modelled.

The most well known area is the image processing done by the brain, measured with single-cell recordings from live brains. Single-cell recordings are done by placing electrodes directly in the brain to record the response of a neuron or a group of neurons to a visual stimulus [37]. For example, certain neurons have been found to respond to edges at a certain 'pixel', that is, a retinal location of the visual field. However, single-cell recordings only give information of one stage in the perceptual process.

According to Goldstein [25], the perceptual process is a chain that goes from receptor stimulus to action through stages such as neural processing, perception and recognition. Laboratory tests can only give information about neural processing and action. Perception and recognition, such as categorizing objects and understanding emotions conveyed by facial expressions, are beyond direct measurements.

In practice, working computer vision systems may partly model human perception, but they are also engineered to exploit whatever a priori knowledge there is available about the application and the problem domain. For example, Ujo Sankari first performs a couple of simple image processing functions, similar in complexity to those performed by the first stages of the human visual system, and then interprets the data under very strict assumptions. Finally, a single pixel of the image is chosen to represent the user and all other data is discarded.

Perceptually interactive systems

The first perceptually interactive system the author knows of was not a game but a digital musical instrument called DIMI-O, developed as early as 1971 by Erkki Kureniemi [34]. DIMI-O featured musical synthesis controlled by the video signal of a video camera. Another early example is the camera based motion tracker system VNS (Very Nervous System) by media installation artist David Rockeby [8]. Rockeby created his first motion tracking systems in 1983 and has been improving them ever since, moving from dedicated hardware to software implementations (softVNS).

QuiQui's Giant Bounce is closely related for example to Perceptive Spaces and other physically interactive story environments developed at MIT Media lab [38][45]. These include several applications where one or multiple human users are tracked. The sensory data is used to control a graphical representation of the user or the reactions of the other characters. Probably the most widely known project is KidsRoom, a large-scale installation where a room is transformed magically into an adventure using several cameras, two background-projected walls and theatrical lighting. In KidsRoom, children can for example pretend that a bed is a boat that floats in a river projected on the walls. The boat is moved by rowing in the air when sitting on the bed. However, the MIT projects do not combine voice and movement in the same way as QuiQui's Giant Bounce does. Voice is used as spoken commands instead of using it to control a character. For example the adventure in KidsRoom starts when a user utters a magic word.

Although the literature at first seems vast, there is little written about designing perceptual computer games and user interfaces for child users in uncontrolled real-life environments, such as homes and daycare centers. Most systems need some adjustments or initialization done by the users or assume that the background is static like in Ujo Sankari. The games developed by Intel and Mattel [18] come close to what we aimed for, but they still have usability defects, such as voice prompts that instruct the user to step aside so that the computer vision can be properly initialized. The games use background subtraction to embed the image of the user into computer graphics, as shown in Figure 3.3. Compared to QuiQui, the games are also designed for the IntelPlay Me2Cam instead of supporting various camera models.

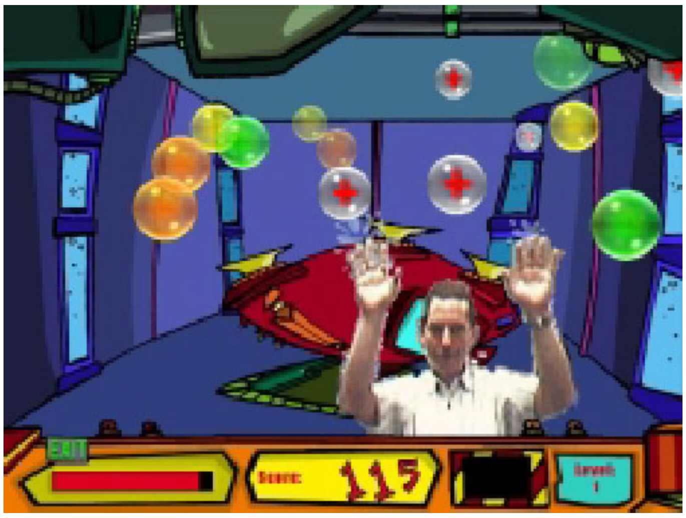

Figure 3.3: An example of the "video avatar" games [18]

Considering game applications, QuiQui's peers can be divided into two categories based on the user's representation in the game. In the first category are the Intel and Mattel style games that use a "video avatar", that is, the video image of the user or a part of it is directly interacting with computer graphics. In addition to Intel and Mattel, similar games have been produced by Vivid Group, targeted for example for amusement parks [1]. Although Vivid Group claims have a patent on the approach, awarded in 1996, the approach was experimented with as early as 1984 in Myron Krueger's pioneering work Video Place [33]. Video Place allowed the silhouettes (monochrome images) of two users to interact with each other and computer graphics, as shown in Figure 3.4.

The alternative to a video avatar is a computer-generated avatar, such as QuiQui. The avatar mimics the user's movements or reacts to them. In general terms, we are talking about computer vision based human motion capture, which has been researched intensively, as seen for example in Moeslund's survey of more than 130 related papers [35]. However, the solutions presented in the

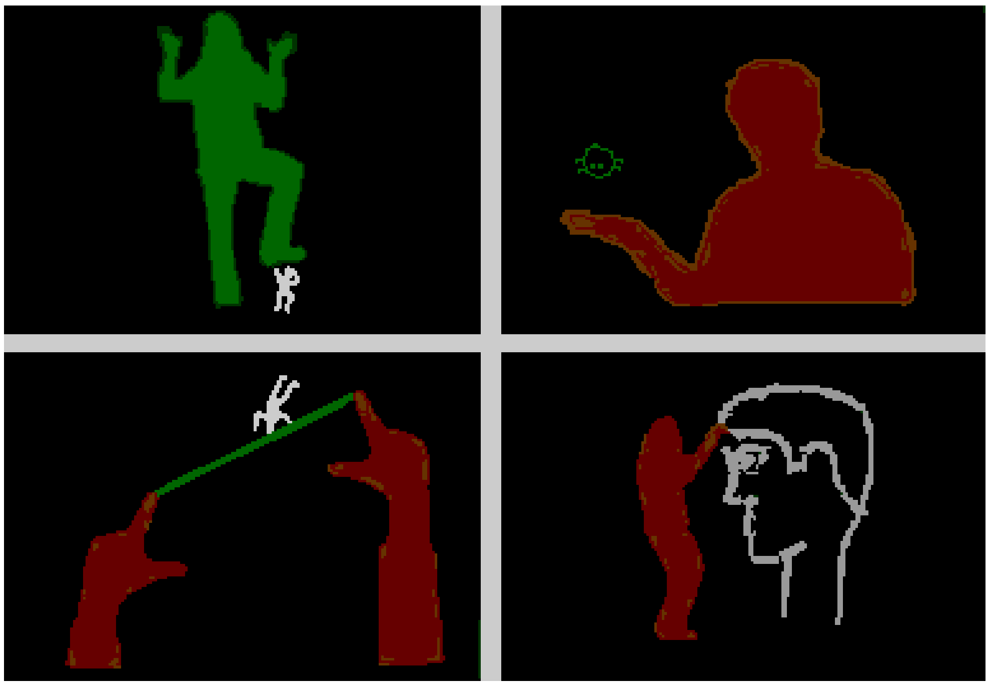

Figure 3.4: Examples of the Video Place by Myron Krueger [9]

literature do not seem to meet the requirements of QuiQui's target group and environment of use. There seem to be no solutions for totally automatic tracking of the user's body in unpredictable environments.

Fortunately, games often restrict the variety of possible movements so that full tracking of the user's body is not needed. For example in Ujo Sankari, the motion is restricted to jumping so that the game only needs an approximation of the position of the user's head. In the scientific literature, Freeman et al. describe several computer vision based game user interfaces that exploit the game context in this way [22][23][21]. One example close to QuiQui is the decathlon game Decathlete, shown in Figure 3.5.

Compared to QuiQui's Giant Bounce, Freeman's experiments used special camera and computer vision hardware instead of general purpose cameras. The user interfaces were also mainly designed for existing games normally played with a traditional user interface, such as a game pad. This is restricted compared to designing the game and user interface together. For example, in games where the avatar jumps when the user presses a button, the height of the jump is usually constant. A perceptual user interface should allow the user to jump to different heights, which requires that more control information is extracted from the user's movements.

Figure 3.5: The Decathlete game user interface by Freeman et al. [21]

3.2 Requirements and assumptions

When designing the user interface for the QuiQui's Giant Bounce prototype, the first step was to define the following set of requirements so that the technical solution would not sacrifice usability and approachability:

1. Completely automatic operation without any learning stages, initialization or settings that need user participation. Our goal is that children can use the game without adult guidance.
2. The methods must adapt rapidly to changes in the environment, including lighting and camera position.
3. The methods must adapt to the differences in various camera models, including frame rates in the range of 15...30 fps, noise, motion blur and color resolution.
4. The system responds with as low latency as possible.
5. The system must tolerate several visible users, either one player and viewers or several users participating collaboratively.
6. The user does not have to wear any specific clothing or markers.
7. As low computational complexity as possible to enable maximal computational resources for the actual game application.

The requirements are based on the experience from Ujo Sankari and the three requirements for human-centered perceptual user interfaces by Crowley et al.: robustness and autonomy, low latency and privacy protection [17]. As Crowley et al. put it, “Usability determines the requirements for technological innovation" Note that privacy protection is not an issue here since it mainly applies to multi-user applications and the video footage of the user is not stored in QuiQui's Giant Bounce.

Requirements 1 and 2 rule out many of the existing computer vision systems, for example all the systems based on an assumption of a constant or almost constant background. This includes the technology behind Intel and Mattel's games [18], most of the MIT Media lab experiments mentioned, several of the experiments by Freeman and also the only webcam based game study the author knows of [39]. Assuming a constant background is lucrative for computer games, because a two-dimensional silhouette of the user can be obtained using background subtraction, as done in Ujo Sankari. Once a silhouette is obtained, simple shape recognition techniques can be applied to produce the control signals needed for a variety of games, as shown by Freeman et al. [22][23][21].

Requirements 1 and 2 are based on the fact that our target group is illiterate and technically incompetent, which makes it difficult for the software to instruct the user. Unexpected changes in the environment do happen when the system is used at homes and the users are children, as D'Hooge points out [18]. Our own usability tests also verify this. Intel and Mattel use voice prompts to guide the user to step away from the camera view for initialization [18], but we find that awkward.

Requirement 3 has severe implications on computer vision methods. Low color resolution, noise, and unpredictable, often automatically controlled color settings such as white balance imply that color based skin and face detectors (see for example Bradski [16]), cannot be used reliably. Fitting curves to image contours (see for example Blake and Isard [15]) is unreliable because of noise and motion blur.

Low frame rates also decrease the accuracy of predictive methods. In computer vision, analysis is often based on predictions, such as the probable area where to search for certain features. The predictions are based on a priori knowledge and the analysis of previous frames. If the frame rate is low or the camera drops frames, as is possible at least in the Windows operating system, the motion of the user may have sudden changes. The motion itself is also not always predictable. For example in a platform jumping game, the user may stay still, waiting for the right moment, and then suddenly bounce in any direction. An example of the amount of motion blur and changes between successive frames is shown in Figure 3.6. The figure contains two frames of a jumping user recorded with a Logitech Quickcam Express webcam in daytime home conditions.

Requirement 4 rules out various gesture recognition methods that recognize movements after they have been completed. Low latency is important for the overall responsiveness of the game. For example, in a game played by jumping, latency can cause the virtual world to feel sticky, as if the avatar was standing in a pool of thick mud.

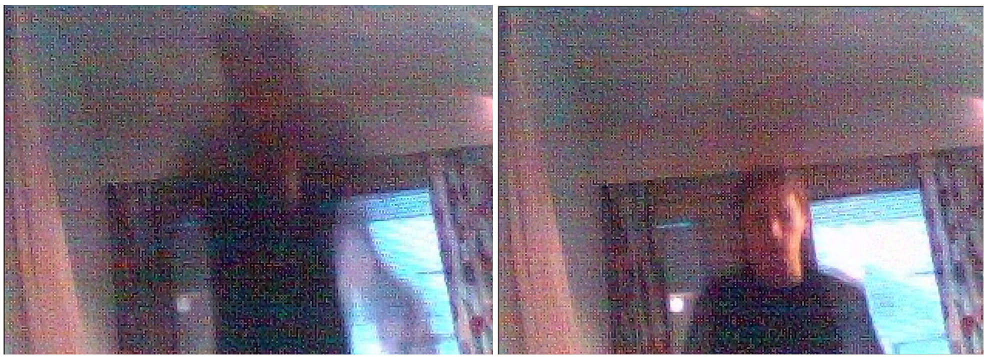
Figure 3.6: Two successive frames captured by a webcam when the user is jumping

Requirement 5 is based on our usability tests and the findings of Inkpen et al. [29] and Stewart et al. [42]. It appears that gaming is a social event for children. It is hard to restrict the situation to one visible user since the others want to watch, standing or sitting beside the main player or crouching on the floor. In our tests, children also liked playing the flying game together with a friend, so collaborative operation should be supported when possible.

Requirement 6 further restricts the computer vision methods. It is defined to enable more convenient collaborative play and the delivery of the game in digital form as a simple downloadable software package.

Adding requirement 7 to all the previous ones, one may wonder what is possible in practice. Fortunately, game applications allow many contextual simplifications. We use the following assumptions:

1. The background is temporally "piecewise" constant. If there are changes, they are either immediate or very slow and gradual. This includes changes in lighting, sudden movements of the camera (if a child for example bumps into the table on which the camera is placed), and changes in the physical environment, like closing or opening a door. According to our experience, it also includes automatic adjustments of exposure and gain done by the webcam drivers, since they usually happen slowly or in discrete steps.

2. The user's movements are restricted. The user faces the camera because the camera is mounted near the screen. Distance from the camera is limited because the user must be close enough to the screen to see the graphics and also because there is a limited amount of space available at

people's homes. Because of the spatial constraints the user's legs do not necessarily fit into the camera view and we restrict the tracking to the upper body of a standing user, a convention also used by Intel and Mattel [18]. The movements are also often restricted to a set specific to the game context, a principle often exploited by Freeman et al. [23][21][22].

3. If there are more than one people visible, the user is not severely occluded by others. This is natural since the user usually wants a clear line of sight to the computer and thus to the camera.

4. Responsiveness is more important than correct tracking of the user. It doesn't matter if the game can be "cheated" using movements other than intended, as long as controlling the avatar is most convenient using the intended movements so that the "right" way of moving is encouraged.

In the author's opinion, assumption 4 is the key to designing practical computer vision and hearing methods for perceptually interactive games. In our tests with children, we have noted that the game experience is generally not degraded if the player can cheat the game. Many times it only challenges the child's creativity. When testing an early prototype of the flying game, a six year old girl started experimenting with different movements. In the interview after the testing she stated: "The best way to get up in the air is to first flap your hands and then you can jump up and down." She was clearly satisfied about her finding. The negative side is that the feedback given by the game can be confusing when the user is learning to play the game.

3.3 Design and testing of the user interface

The following treatment is focused on the *human-avatar interface*, the mapping of the user's actions to the avatar's actions and the technology behind this. The overall game design and the *avatar-virtual interface* are beyond the scope of this paper. The term avatar-virtual is used here to denote the means of interaction between the avatar and the game world, the design of which is closer to user interface and usability design in general, including metaphors, logic, information design etc.

In the context of computer games, the definition of usability is rather vague (see for example the discussion in ACM CHI-WEB archives [10]). The basic contradiction is that a good user interface tries to make things easier for the user, but computer games are played largely because of the challenge they present. Games are not software tools used to get things done effectively, although they can contain tool-like parts, such as the main menu of the prototype. Our view is that the human-avatar interface should be as intuitive and transparent as possible so that it lets the user think and act from the point of view of the avatar. It certainly is a usability issue, if the user, as his or her real-world self, does not know how to move so that the avatar flies to a certain direction. On the other hand, the avatar's interaction with the game world can be challenging. It is not necessarily a usability problem, if the user, as the avatar, must struggle to jump high enough to get over an obstacle in the game.

Note that the division into human-avatar and avatar-virtual interfaces is not clear when considering game physics. Game physics are the general laws controlling the avatar's movement in the virtual world. The game physics largely define the overall feel of the user interface, described with adjectives like light, heavy, fast, slow etc. Variables such as gravity have an effect on how the avatar reacts to the user's movements. However, they also have an effect on how the avatar interacts with the game world. Properly designed game physics should increase the feeling of being in control of the game. On the other hand, in games like Liero [11], the game physics constitute the main challenge and enjoyment of the game. The physics are adjusted so that the game requires lightning fast reflexes and decisions when the avatar is flying through the air or swinging at the end of a rope.

In general, it seems that the human-avatar interface is designed to be as simple as possible, with the exception of games like Tomb Raider [4] or Die By The Sword [12]. Those games offer increased realism with a more complex and expressive interface that enables actions such as somersaults or separate control of the avatar's upper and lower bodies.

QuiQui's Giant Bounce uses simple sets of movements that vary according to the game task (sub-game). The user interface design is based on the game design and narrative. In the flying game of the prototype, the basic setting we started with was that QuiQui has to help a yellow watering can to water the dry desert. This is done by flying through the clouds in the sky to make them rain. QuiQui holds a pair of big leaves in his hands so that they act like wings. The game starts with a metaphorical instruction spoken by the watering can "Flap your hands like a bird to fly up in the sky".

3.3.1 Methodology: usability testing with functional prototypes

The basic design method used was to experiment with technology and produce a simple, but interactive prototype. The prototype was then tested with children. If the prototype did not work, the children were observed to find out how they tried move to use the prototype. If a certain way of moving was tried frequently, as in the case of the flying game, it was considered to be the most intuitive way of moving for the children. The interaction and technical design was then modified to match it.

Among others, Druin [19] and Hanna et al. [26] have written about collaborative design and usability testing with children. Druin et al. present a method called Cooperative Inquiry for incorporating children in the design process, for example through collaborative low-tech prototyping sessions. Hanna et al. give guidelines for adapting conventional usability testing methodology for children.

Since we were interested in aspects such as the intuitiveness of the user interface, we adopted an approach of iterative usability testing and interviews with users that are new to the game. Druin suggests a long-term relationship with a group of children [19], but in our case such an approach would be more fruitful in overall design of the game or for example the story and character design.

The main testing method used was peer tutoring, a method for children's education adapted for usability testing by Johanna Höysniemi [27]. The test was started with the metaphorical instructions and with instructions to fly through all the clouds. After that the test subjects tried to learn to fly and they were instructed only if they were not successful or seemed frustrated. After a test subject had played the game a couple of times, it was his or her turn to teach the next child to play. We used two approaches, one where one child tutor instructed one child tutee and one where two child tutors instructed one child tutee. The peer tutoring approach was used because we wanted the children be able to teach other children how to use the game, which is important in social settings, such as schools.

The tests were conducted at homes, and at a combined preschool and school. The tests were videotaped to record the movements and gestures of the users. The tests were done using perceptually interactive prototypes, initially with more simplifications than in the final version. Although there has lately been some research on prototyping tools for perceptual user interfaces [40], they are more suitable, for example, for user interfaces where the user points at things. In skill-and-action games such as the flying game, the interaction is more fast-paced and prototyping is more difficult.

Two iterations of the user interface were produced based on two tests with 12 and 16 children. The subjects were of ages 5 to 9. In addition to the second test, the final prototype has been tested informally with several users. Informal tests were also used between the two usability tests to try out various technical implementations of the second and final user interface. We also had feedback from teachers and parents who downloaded and installed the game.

3.3.2 Results of the testing

We evaluated the different iterations and technical implementations of the user interface based on how quickly the users were able to fly through all the clouds. Only the human-avatar interface was changed between the tests. Other aspects of the game, such as the locations of the clouds, were not modified. The results of the second test indicated that all the children were able to play the game and the average playing time dropped to 66% of the first test. All children learned to fly and could fly through all the clouds. The children also wanted to play the game several times, which indicates that they liked the game. In informal tests we also found out that the game is suitable for at least some 4-year-old children.

Interaction model

In addition to the conclusive results, the tests provided significant information during the design process. In the first test, the most frequent reaction to the metaphorical instructions was that the children waved their both hands to fly and bent their upper bodies sideways to control the direction of the flying. In the first prototype our initial suggestion was that the user would use only one arm to fly sideways. This was changed to match the observations in order to make the user interface more intuitive.

Technical design

From the point of view of technical design, the main point learned was that the variety and flexibility of children's movements are very difficult to anticipate, even though certain ways of moving were more frequent than others. This directed the computer vision methods towards analyzing holistic and rather vague image features such as image moments instead of precise tracking of the user. Examples of the various ways the children moved are shown in Figure 3.7.

Spatial design: the Magic Square

We also encountered problems with the spatial design of the user interface. Because of the limited space available at schools and especially at homes, it is difficult to place the camera so that the user is always visible. Instead, the user must be guided to stay within the view of the camera. This is problematic especially when an exact one-to-one mapping between the user and the avatar is not possible. When the avatar is flying in the air, there is no intuitive mapping for the user moving sideways. Because of this, there is a gap in the feedback of the user interface, and the user can accidentally step out of the camera view. This happened several times in the usability tests.

The problem was initially tackled with the small camera window in the bottom left corner of the screen. It helps in verifying that the user is positioned correctly, but we found out that it was hard for the test users to monitor their position. It appears that the camera view is mainly helpful for adults in determining the correct placement of the camera. In general, the younger the users, the easier they forgot that they should stay near a certain optimal location.

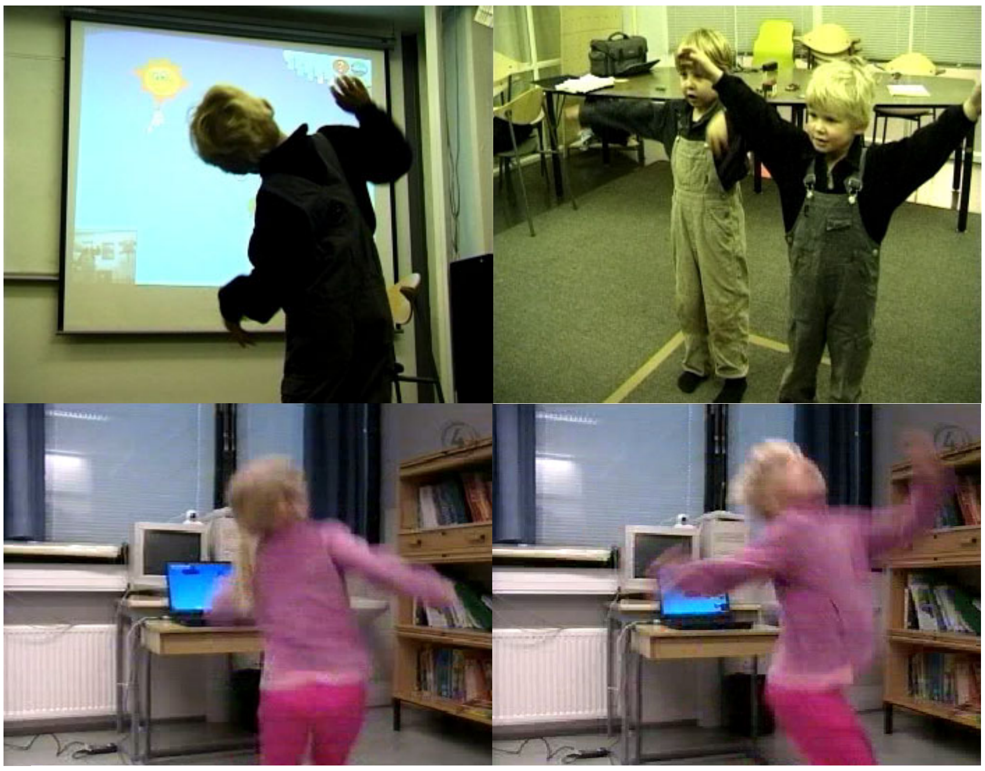

Figure 3.7: Test users learning to fly. Note that the figures at the bottom are blurred because of the rapid movements of the user

We did not want to abandon the concept of a flying avatar. Also, we did not want to set any additional technical constraints for the game, such as a specific camera model with a wide enough field of view. Fortunately, Johanna Höysniemi came up with the idea of a "magic square" marked on the floor with marker tape. Our tests show that this considerably helps the children to remember the spatial constraints. They seemed to understand it easily when we explained that "you must stand in the magic square so that the eye of the computer (the camera) can see you." This instruction was usually the first one given when the children explained each other how to play the game.

Although adults might regard things like magic squares as cheap compromises or cheating, ‘magic’ seems like a useful concept when designing children’s technology. It is important and natural in children’s fantasy and play. Another good example of using magic to overcome difficult design constraints is given by Druin [20]. She describes how hand-held “magic keys” were designed for a collaborative finger painting system. The system required that each user touches a device that identifies him or her. The technology manufacturer’s original solution was that each user sits on a special pad while drawing, which was not natural for the children participating in the design.

To summarize, one may expect space-related problems if the avatar and the user move differently and also because of limited playing space and the field of view of the camera. The magic square is not necessarily the right solution for all games. For example, in a platform jumping game, a direct mapping between the user’s and avatar’s horizontal positions could be used, but it would confine the avatar inside a certain area. One solution is to map the location of the user to the avatar’s velocity, that is, to map physical variables to the derivatives of their virtual counterparts. This approach was used in Ujo Sankari as well as the MIT Media Lab SURVIVE system [45]. In SURVIVE, the user plays the game Quake with a motion tracked mockup of a gun, standing in front of a large screen.

Cognitive load and feedback to the user

We also encountered problems that were apparently caused by the high cognitive load of the user interface. The main observation was that it is hard to get the user’s attention when he or she is learning to fly. The user is also further away from the screen than in traditional computer games, so that the graphics must be scaled, making the avatar larger in relation to the size of the screen. This complicates things as the user has less time to react when flying around. Although we tried to keep the visual appearance of the game as simple as possible, there was so much happening that the test users did not notice everything. For example, the users did not always notice the raindrop symbols in the top right corner of the screen.

In the prototype, we have not yet found any general solutions to provide feedback that would catch the user’s attention inevitably. It seems that rather extreme techniques might be needed, such as cinematic close-ups of important events.

3.4 Technical design and implementation

This section describes and evaluates the technical design and implementation, including the game physics, computer hearing and computer vision. The design is an example of a simple and practical technical solution that relies on the requirements and assumptions presented.

The general design principle is to use as low-level features of the image and sound data as possible, considering the game context. An example of a low-level image feature is the center of mass, in contrast to a high-level interpretation such as three-dimensional pose of the user.

The more decisions and interpretations are made, the more there are chances for errors. Since the environment of use and the actions of the child users are unpredictable, it is difficult to ensure that the assumptions behind the decisions and interpretations hold in every situation. A simple motion tracker that interprets various movements as flying is better than a sophisticated tracker that causes the avatar to fall down when it makes a false interpretation and moves the avatar's legs instead of its arms.

The treatment is more technical than the preceding chapters so the less technically oriented readers may want to skip the mathematical formulae. The principles of the solutions can be understood without mathematics, but the formulae are provided for the sake of completeness.

3.4.1 Computer hearing

The computer hearing is the simplest part of the technology since the game only needs to detect whether the user is shouting or not. In practice, the prototype detects all loud sounds as shouting, but it does not matter because falsely detected shouting has no negative consequences in the game. This is in agreement with assumption 4 presented in Section 3.2.

The main benefit of the approach is computational simplicity. Children also seem to have fun experimenting with different sounds. In one test with three children, a boy found out that he could make QuiQui breath fire by clapping his hands. After that, the other children tried it out enthusiastically. According to our tests, especially young boys are excited about the voice interface.

Although the voice input mechanism is simple, it seems to increase the excitement of the game and support the illusion of being a dragon. Because of the direct mapping between shouting and breathing fire, the voice interface does not alienate the user from the game. The interface allows the user to express himself or herself as the avatar, as opposed to voice interfaces that recognize commands spoken to the game.

Technically speaking, shouting is detected by filtering and thresholding the microphone input signal power. Simply detecting large input values is not enough, because the game should adapt automatically to different levels of background noise. The first step is to compute the signal power in each 50ms window of audio samples delivered by the sound card's capture driver:

$$P(m) = \sum_{i=1}^{N} x(n)^2, \tag{3.1}$$

where m is the window index, N is the window length in samples and $x(n)$ is the input signal. Since we are only interested in the signal power relative to the power of the background noise, we filter the power using a first order IIR highpass filter:

$$P_{hp}(m) = \frac{1}{1+\omega}P(m) - \frac{1}{1+\omega}P(m-1) + \frac{1-\omega}{1+\omega}P_{hp}(m-1), \tag{3.2}$$

where ω is the cutoff frequency of the filter in the range 0...1. The filter makes the game adapt to different levels of background noise. If the power of the background noise stays constant or changes only slowly, the filtered power $P_{hp}(m)$ is close to zero. When the user begins to shout, $P_{hp}(m)$ peaks and begins to decay exponentially as shown in Figure 3.8, with $\omega = 0.1$. A threshold value T is used so that the avatar breathes fire as long as $P_{hp}(m) > T$.

Treatments on the mathematical theory of digital filters can be found in various digital signal processing textbooks (see e.g. Ifeachor and Jervis [28]). However, the filter used is simple and can be understood with common sense. Consider for example the case when $\omega = 1$. In that case the last term $\frac{1-\omega}{1+\omega}P_{hp}(m-1)$ of Equation 3.2 becomes zero and the filter's output is determined by the first two terms. The output is simply the difference of the last two input samples, scaled with 1/2. If the input signal changes to a constant value, the filter's output peaks for one sample and then stays at zero. Now when ω is decreased towards zero, the last term begins to have an effect on the signal. It is an integrator term, because it adds the previous value of the filter's output to the current output value. In effect it acts as an opposing force to the first two differentiation terms, trying to increase the magnitude of the output if the input stays constant. When $\omega = 0$, the differentiation and integration terms are 'equally strong' so that $P_{hp}(m)$ stays equal to $P(m)$ if they are equal initially.

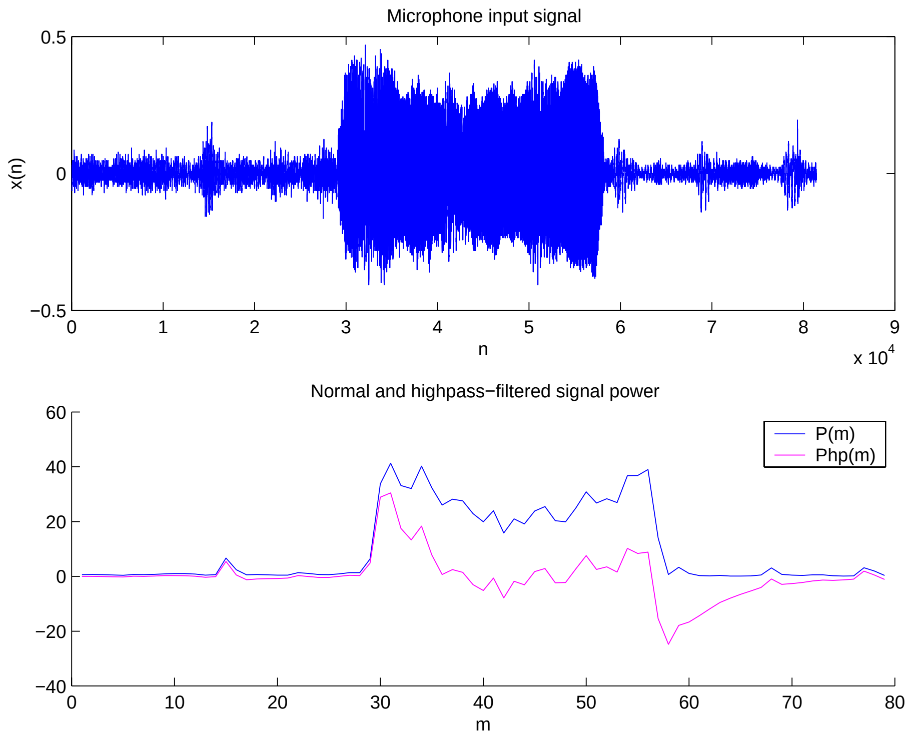

Figure 3.8: The operation of the digital filter used in the computer hearing

3.4.2 Game physics

The game physics for the avatar had to be designed before the computer vision, since they determine the control signals that need to be computed from the user's movements. The solution uses basic Newtonian mechanics, although interpreted rather loosely. When designing game physics, the real laws of physics are merely exemplary conceptual tools.

QuiQui is modelled as two masses connected by a weightless rod as shown in Figure 3.9. Instead of the center of mass, the moments caused by the forces F and G are computed relative to the origin which acts as a virtual point of support. The force G denotes gravity and F denotes the "thrust" force caused by the user's movements. The angle β between the vector F and the rod equals the angle between the user's spine and the vertical axis. This allows the user to control the rotation of the avatar by bending sideways when flying. The mass at the lower end of the avatar is further away from the origin so that it causes the avatar to rotate back to upright position if the user is not moving. Note that QuiQui is not flexible like the user. The animation simply has three frames with hands held up, down and straight to left and right.

To update QuiQui's position and the angle α, the game needs to compute QuiQui's velocity and angular velocity. These are computed from the well

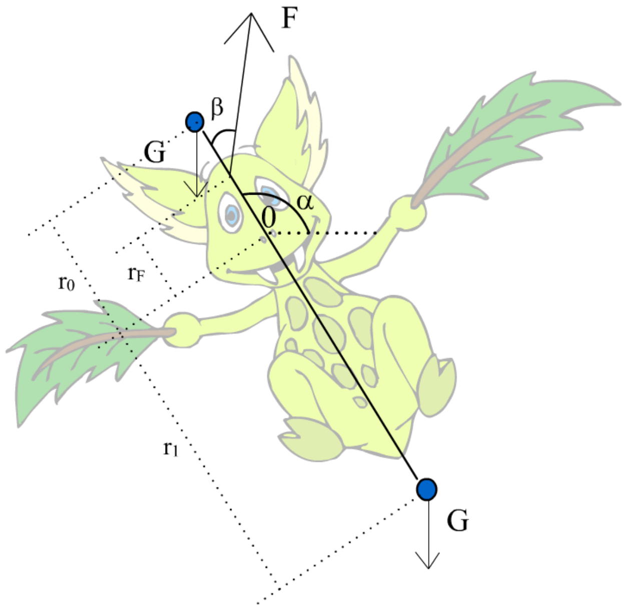

Figure 3.9: Physics of the flying avatar

known equations found in any physics textbook. The velocity is solved from the force equation

$$\sum \mathbf{F} = m\frac{d\mathbf{v}}{dt}, \tag{3.3}$$

or with the x and y components separated,

$$\begin{aligned} \sum F_x &= m\frac{dv_x}{dt}, \\ \sum F_y &= m\frac{dv_y}{dt}, \end{aligned} \tag{3.4}$$

that is, the sum of forces equals mass times acceleration which is defined as the time derivative of velocity. The angular velocity ω is solved from the moment equation analogous to the force equation,

$$\sum M = J\frac{d\omega}{dt}, \tag{3.5}$$

that is, the sum of moments equals the moment of inertia times the angular acceleration which is defined as the time derivative of angular velocity.

The physics are simulated by approximating the differentials with differences of the variables between successive simulation steps. In the case of Equation 3.3, $d\mathbf{v} = \mathbf{v}(n) - \mathbf{v}'(n-1)$ and $dt = t(n) - t(n-1)$, where n is the index of the simulation step. In the prototype, the simulation is carried out once for each frame of the graphics. If F, m, the previous velocity $\mathbf{v}'(n-1)$ and the time between successive simulation steps are known, the current velocity $\mathbf{v}(n)$ can be solved from Equation 3.3.

The sum of forces is simply computed as

$$\begin{aligned} \sum F_x &= F\cos(\alpha+\beta), \\ \sum F_y &= F\sin(\alpha+\beta) - 2G. \end{aligned} \tag{3.6}$$

However, computing the moments is not done 'by the book'. A moment caused by a force equals the force times the distance of the force vector from the rotational center. Considering the mass at the lower end of the rod in Figure 3.9, its moment equals $M_1 = Gr_1\cos\alpha$. The moment is zero when QuiQui is in an upward position and increases when QuiQui rotates, reaching its maximum when QuiQui flies horizontally. However, this was found to be too restrictive if the user wants to fly upside down, so the moment is actually computed as

$$M_1 = \min(d_{max}, \cos\alpha) G r_1 f_M, \tag{3.7}$$

where d_{max} is an arbitrarily chosen constant, 0.2 in the prototype. The moment is clipped to a maximum value before QuiQui reaches a horizontal position. The sum of moments is then expressed as

$$\sum M = G f_M (r_1 - r_0) + \mathrm{sign}(\beta) F s, \tag{3.8}$$

where s is the distance of the vector F from the origin. Computing s requires some basic trigonometry, resulting in

$$s = \frac{r_F\left[\sin\alpha - \tan(\alpha+\beta)\cos\alpha\right]}{\sqrt{\tan(\alpha+\beta)^2+1}}. \tag{3.9}$$

In addition to the user's movements, the velocity and angular velocity are affected by the game environment. After solving the velocity and angular velocity from the force and moment equations, they are damped by multiplying them with damping coefficients in the range 0...1. This roughly simulates the air resistance of the game world.

To add a little bit directional stability, the velocity vector is rotated in the direction of the angular velocity using the usual two-dimensional rotation equations [32],

$$v'_x(n) = v_x(n)\cos\alpha_{rot} + v_y(n)\sin\alpha_{rot} \tag{3.10}$$

$$v'_y(n) = v_y(n)\cos\alpha_{rot} - v_x(n)\sin\alpha_{rot}. \tag{3.11}$$

The rotation angle is computed as

$$\alpha_{rot} = f_{rot}(\alpha(n) - \alpha(n-1)), \tag{3.12}$$

where f_{rot} is the rotational coefficient, 0.175 in the prototype. The rotation makes QuiQui behave a bit more like a boat in the water as opposed to the simple masses and rod flying in the air, which feels more natural, at least in the author's opinion.

3.4.3 Computer vision

To make the system robust to changes in the environment, the basic approach of the computer vision is differential motion analysis [41], based on intensity differences between consecutive video frames (temporal difference). The intensity differences are strongest at the parts of the frames containing motion. Changes in lighting etc. cause only temporal disturbances in the system, opposite to the background subtraction used in Ujo Sankari. The same approach has been used by Rockeby in various VNS versions [8].

The drawback of using only the temporal difference data is that only the moving parts of the user can be detected. Fortunately, the analysis is simplified by the assumption that the movements of the user are limited to those relevant to the game context. The system assumes that if the user is not simply standing still or walking around, he or she is trying to fly.

In short, the computer vision of the prototype combines simple edge-based shape classification with feature extraction based on image moments. The shape classification is used to decide whether the user is flying or not. If the user is flying, image moments are used to compute the angle of the user's upper body.

The analysis consists of the following four processing stages, operating on 8-bit grayscale images:

1. Reduction of resolution to 120x96 pixels with spatial averaging (lowpass filtering) to reduce noise. The intensity of each output pixel is the average of an 8x8 pixel area of the input. The video is originally captured at 320x240 pixels or the closest resolution supported by the camera drivers.
2. Temporal differencing. Each output pixel is the absolute value of the difference of the corresponding input pixels of the last two input frames.
3. Thresholding. A threshold value is determined and pixel values below it are considered as noise and set to zero.

4. The image and the trajectory formed by its topmost nonzero pixels are analyzed to determine the output variables used to update the game physics.

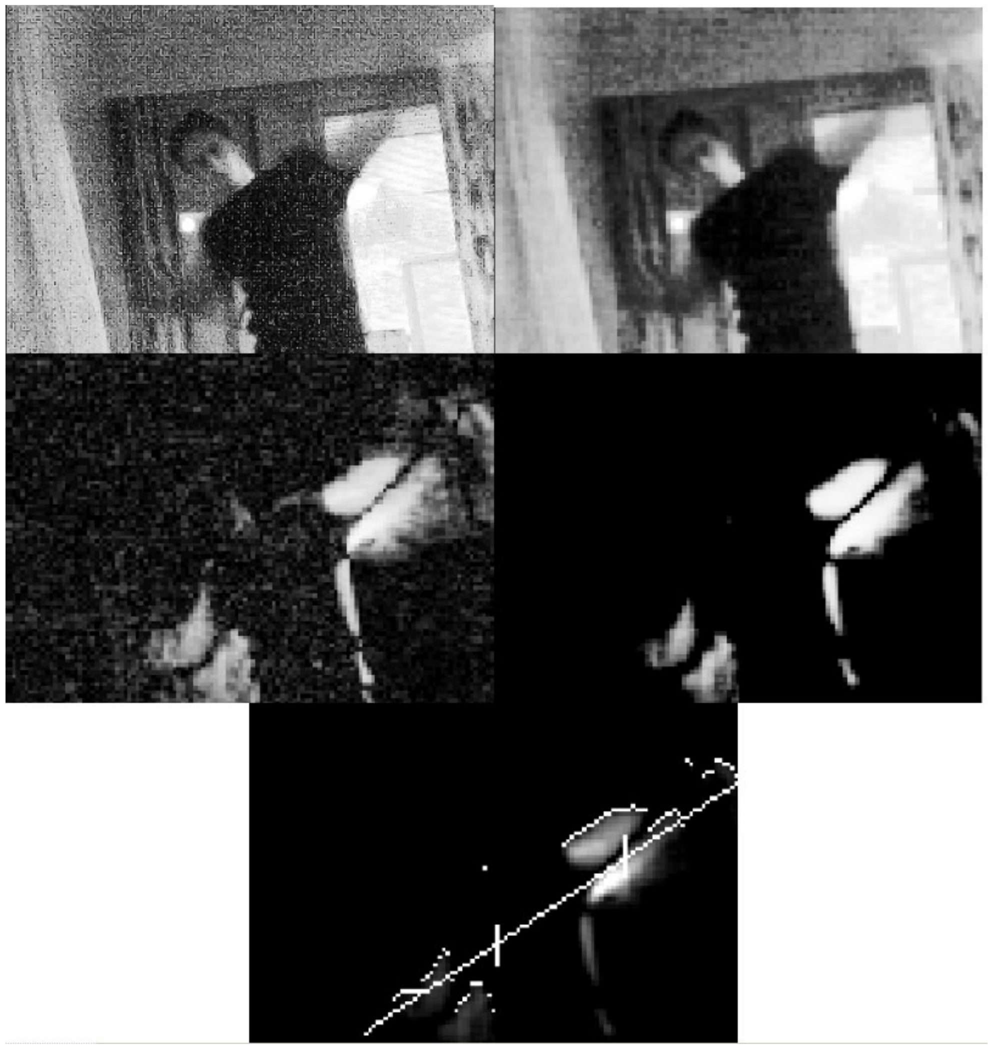

Figure 3.10: The original input and the four processing states of the computer vision (from left to right and top to bottom)

Figure 3.10 shows an example of the camera input and the outputs of the processing stages. In the last image, the pose of the upper body is visualized as a straight line and the topmost nonzero pixels are plotted with maximum intensity. Note that the brightness of the images has been increased for better printing quality.

The threshold value of stage 3 is determined using histogram-based thresholding, various approaches to which are described by Sonka et al. [41]. A histogram of the intensity values of the temporal difference is computed, as shown in Figure 3.11. The figure shows two examples of temporal difference images with their histograms plotted at the bottom. The histogram is an array where the element h_i equals the number of pixels with the intensity i. When there is no movement, the histogram only shows the distribution of the temporal difference of the background noise, seen as a narrow peak in the left side of Figure 3.11. When the user moves, the histogram becomes wider and has several local maxima and minima. The threshold is determined as the first local minimum of the histogram when searching from the maximum towards greater intensity values. According to our tests in various environments with a total of seven different camera models, this seems to work sufficiently well. Note that the images in Figure 3.11 are normalized for better printing quality. Especially the left side image contains only low intensities and would otherwise appear as a black square.

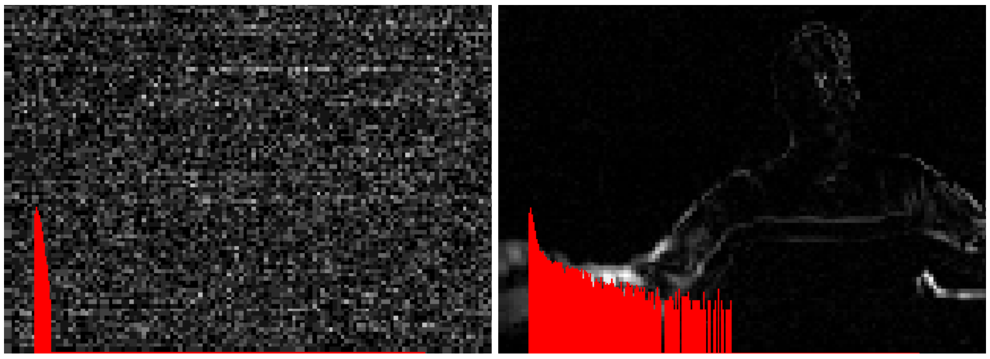

Figure 3.11: Computing the histogram from the temporal difference: histogram of noise with no movement (left) and the histogram with a user moving in front of the camera (right)

The fourth stage of the processing is carried out in three steps. The first step is to detect whether the user is doing something or standing still. The criterion used is the number of nonzero pixels. The analysis is continued only if the amount exceeds an experimentally determined value, currently half of the image width.

The next step is to detect if the user is trying to fly instead of simply walking around. A parabola is fitted to the topmost pixels of the image in least-squares sense, as shown in Figure 3.12. When the user is walking, the parabola usually fits well and its peak is located near the head of the user. The analysis is not continued if the quadratic coefficient of the parabola is negative enough and the peak of the parabola is horizontally near the center of the detected topmost pixels.

The analysis of the angle β and force F in Figure 3.9 is based on image moments. Image moments are statistical characteristics that have been used for computer vision based game user interfaces by Freeman and Sengupta et al.

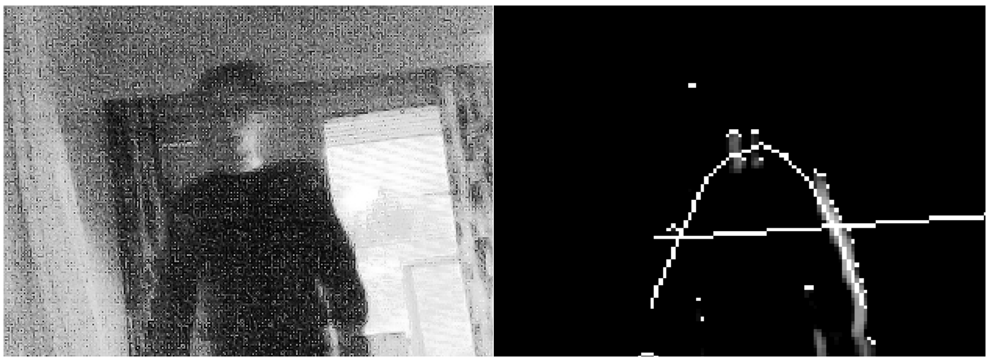

Figure 3.12: Detecting a walking user

[23][21][39]. An image moment of order (p+q) is computed as the summation [41]

$$m_{pq} = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} x^p y^q i(x,y), \tag{3.13}$$

where i(x,y) is the pixel intensity at coordinates x and y.

For user interface purposes, moments up to second order can be computed from the silhouette of the user to obtain estimates of the size, location and orientation of the silhouette [23][21]. Freeman et al. also suggest that they could be computed from the temporal difference [21]. However, in this case the orientation information obtained from the moments is ambiguous, depending on the parts of the user that are moving. The principal axis of the difference may appear to be in the direction of the user's hands as well as the user's body.

Our solution for determining the pose of the upper body is to compute two spatially biased mass centers from the thresholded temporal difference. We use a left and right mass center, (x_{left}, y_{left}) and (x_{right}, y_{right}), marked with short vertical lines in the last phase of Figure 3.10. The orientation of the user's shoulders is approximated with a line drawn through the centers, also shown in the figure. The coordinates of the mass centers are computed from moments up to third order, by weighing the pixels differently depending on their position:

$$\begin{aligned}
x_{right} &= \frac{1}{k_{right}} \sum_{x=1}^{w} \sum_{y=1}^{h} x(x^2+y)i(x,y), \\
y_{right} &= \frac{1}{k_{right}} \sum_{x=1}^{w} \sum_{y=1}^{h} y(x^2+y)i(x,y), \\
x_{left} &= \frac{1}{k_{left}} \sum_{x=1}^{w} \sum_{y=1}^{h} x((w-x)^2+y)i(x,y), \\
y_{left} &= \frac{1}{k_{left}} \sum_{x=1}^{w} \sum_{y=1}^{h} y((w-x)^2+y)i(x,y), \qquad (3.14)
\end{aligned}$$

where h and w are the height and width of the image in pixels. The y coordinates start from 1 at the bottom of the image, growing towards the top. The scaling factors k equal the sum of the weights,

$$\begin{aligned}
k_{right} &= \sum_{x=1}^{w} \sum_{y=1}^{h} (x^2+y)i(x,y), \\
k_{left} &= \sum_{x=1}^{w} \sum_{y=1}^{h} x((w-x)^2+y)i(x,y). \qquad (3.15)
\end{aligned}$$

The formulae can be understood comparing them to the usual way of computing a mass center, that is,

$$\begin{aligned}
x_c &= \frac{1}{k} \sum_{x=1}^{w} \sum_{y=1}^{h} xi(x,y), \\
y_c &= \frac{1}{k} \sum_{x=1}^{w} \sum_{y=1}^{h} yi(x,y), \\
k_c &= \sum_{x=1}^{w} \sum_{y=1}^{h} i(x,y). \qquad (3.16)
\end{aligned}$$

Compared to this, the left and right mass centers are computed by adding the weight factors (x^2+y) and $((w-x)^2+y)$. Pixels with greater x-coordinates are given more weight in the summation of the right mass center. The pixels closer to the top of the image are also given more weight in both mass centers to reduce the effect of the movement of the user's lower body.

The position of the hands can be impossible to detect accurately because of motion blur, as seen in the top right corner of Figure 3.10. If the user's hands are moving fast enough, they can be perceived as an almost constant area of movement in the temporal difference. If flying is detected, the hand position information is changed by one animation frame according to the vertical movement of the mass center of the temporal difference. According to our tests this works sufficiently well.

Because of the ambiguity of the hand position information, the thrust force F in Figure 3.9 cannot be computed from the angular velocity of the user's hands, although this would probably be the most intuitive interpretation. Instead, it is directly proportional to the amount of movement, measured as the amount of non-zero pixels. This is clearly a weakness, because it makes the system sensitive to changes in camera optics and the distance of the user. However, considering a single setup of the game, these properties vary only a little. The user can adjust the overall sensitivity of the game in the main menu, which would be a useful feature even if the force could be determined more accurately.

3.4.4 Evaluation of the technical design

The computer hearing and game physics seem to work well. They did not cause any problems in the tests and the physics seem to allow quite natural movement in the air. The computer vision is clearly the most complex and challenging part of the design.

Because of the manual adjusting of the sensitivity of the computer vision, requirement 1 presented in Section 3.2 is only partially fulfilled. The game works with the default sensitivity, but in some cases it may feel too light or heavy.

Requirements 2, 3, 4 and 6 are met because of the use of temporal differencing and low-level image features. The system analyzes all movement, regardless of shape, color etc. The latency of the system is only one video frame.

Considering requirement 5, the presence of other people than the user does not affect the game as long as they move less than the user so that the user dominates the analysis. However, the game is sensitive to other movement when the user wants to fall down and just stands still.

The system also supports collaborative play because the motion analysis based on image moments is vague enough to treat several users as one. However, this requires that the users are standing close enough to each other and flying in the same direction.

Considering requirement 7, the computer vision is computationally efficient, leaving most of the processing power of current personal computers for the actual game. The implementation caused an increase of 10% in the CPU load of our test system, compared to a bypassed situation, where a video stream was captured and converted to grayscale, but not fed through the processing stages described in this paper. The test system used was a 1 GHz Pentium III Dell Inspiron 8100 laptop computer capturing video at 30 fps using a Creative Video Blaster Webcam 5 camera.

Note that the system depends heavily on the assumption of the user facing the camera, as stated in Section 3.2. In fact, the system assumes that the user's upper body and hands move in a plane perpendicular to the camera so that their pose can be analyzed from the two-dimensional temporal difference images using the mass centers. According to the usability tests, the assumption holds in general, but in some occasions the children bent their bodies also towards the camera, as shown in Figure 3.13. Although flying sideways becomes difficult in this case, the problem is not fatal. After trying for a while, all children learned to control the game correctly.

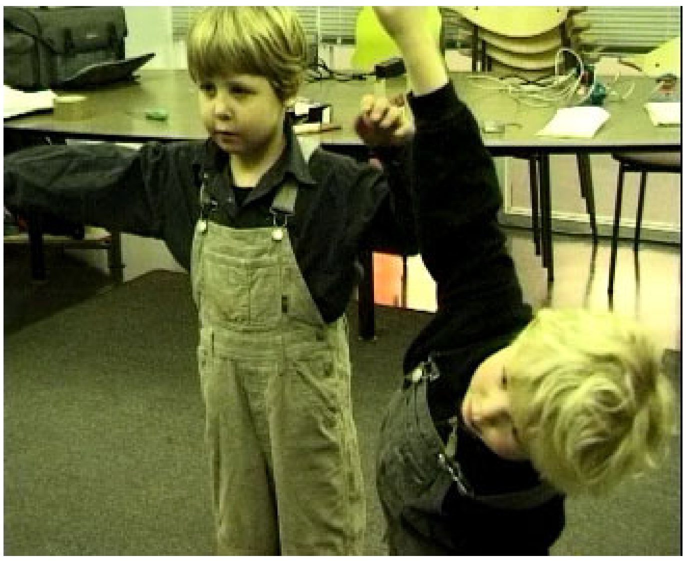
Figure 3.13: Bending towards the camera instead of bending sideways (front). Note that both children look towards the camera and the monitor.

To summarize, the computer vision solution has been proven to work and enables an enjoyable game experience, although it is not perfect according to the requirements defined. The main benefits of the approach are robustness and simplicity. The main drawback is the dependency on the game context, that is, the loss of generality. However, parts of the system can be also applied to other user interfaces. For example, it seems generally useful to recognize shapes from the edges of the temporal difference to derive data or conclusions for other analysis and processing stages. Although only the moving parts of the user are visible in the temporal difference, some movements can be easily recognized.

3.4.5 Dead ends: adaptive background models and active contours

In the beginning of the project, other approaches were also tried. Although background subtraction could not be used as such, adaptive background subtraction methods were tried. It seemed that the image could be divided into the background and the silhouette of the user as in Ujo Sankari, provided that the model of the background have some intelligence. It should, for example, notice if a static object, such as a chair, is moved. This could cause a short disturbance in the system, but after that the system should be able to treat the chair as background again.

There are several adaptive or statistical background subtraction methods described in the literature, of which Toyama et al. give a comprehensive comparison when describing their Wallflower system [44]. The author experimented with various methods, but the results were not good enough. However, the author did not have time to implement and test all the methods. In general, the source code or binary executables of the various systems described in computer vision journals and conferences are rarely available and if they are, they are often made for high-end hardware, such as Silicon Graphics workstations.

It seems that the fundamental problem of background modelling is that pixel-by-pixel processing is not enough for user interface purposes. Background modelling and subtraction is perhaps most often applied in surveillance systems with a static camera. In surveillance image data, the background is only occasionally covered by moving objects. This makes it possible to use a pixel-wise statistical model, such as the median value of the intensities observed in previous frames [41]. In QuiQui's Giant Bounce, the user moves mostly within a small area so that the background is revealed only occasionally, which confuses the statistical models.

It seems that a working background model should combine several cues, including pixel-wise statistics, motion and some model of the user. Although the Wallflower system takes a step in this direction by combining pixel, region and frame level processing, no perfect system exists to the author's knowledge. Background modelling for perceptual user interfaces remains a topic of further research, since a working background model would make a versatile base on which different user interfaces could be developed. In addition to that, camera manufacturers should provide more comprehensive support for the Microsoft DirectX camera control interfaces. The automatic gain and white balance controls of several webcams complicate background modelling. In principle, they can be switched off programmatically using the DirectX API, but not all cameras support that in practice.

Tracking of the user using active contours was also experimented, based on the book by Blake and Isard [15], but working solutions were not reached. Active contours are deformable shape models that are attracted to image features, such as strong edges. In short, the algorithms presented by Blake and Isard initialize an active contour to the outline of some object, such as the hand in Figure 3.14. After the initialization, the scaling, rotation, translation and other parameters of the contour are updated so that it is attracted to strong image features found on a number of normals of the contour. The main strength of the approach is that searching the features along the normals is computationally quite efficient. Matching a model to all possible locations in an image is not possible in real time, especially if the model has several degrees of freedom, such as rotation and scaling in addition to translation.

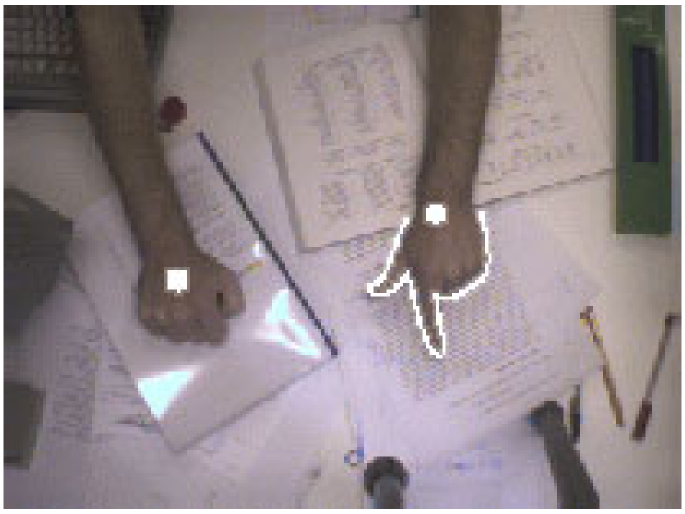

Figure 3.14: Tracking a hand using active contours [13]

Initially, active contours seemed as a very promising tool for game user interfaces. The main problems encountered were the motion blur in some cameras and the complexity and flexibility of the human body. Although it was easy to track the user's head using an ellipse, the tracking sometimes failed when the user made rapid and unpredictable movements. The ellipse could then be stuck to other image features, such as background objects or the user's hand, which would be disastrous in a game situation. The author initially reasoned that all that would be needed was a more complex model, adding hinged extensions for the user's shoulders and hands. It turned out that this only made things worse, since the model had more degrees of freedom and could fit more easily to wrong places in the image. It is also very difficult to design a precise model, considering the variety of children's movements, shown in Figure 3.7.

Note that the decisive factor against using active contours was the motion blur of several low-cost cameras. With sufficiently high quality cameras, active contours could work in game user interfaces, at least as a part of a larger computer vision system, for example when used together with background modelling.

4 Conclusions

This thesis described the concept and interaction design of a perceptually interactive computer game for children. The thesis also described a working prototype that implemented the relevant ideas, including a multimodal perceptual user interface and the underlying computer vision and hearing technology. The prototype won the Pikku Kakkonen category in the mindTrek 2001 competition (best children's multimedia) and it was selected among the 14 winners of the Milia New Talent competition 2002. The project has also had a lot of positive publicity in the media, including several newspaper articles and two appearances on national television.

The computer vision of the prototype combines several methods from the scientific literature with original work, such as using third order image moments and classification of motion based on the contours of thresholded temporal difference. The technology is designed from the point of view of usability. The design is based on a set of requirements and assumptions presented, defined considering the target group and environment of use.

The target group was incorporated in the design via iterative usability tests and interviews conducted with children. In general, the prototype was found to fulfill the defined requirements, although it is not perfect. The computer vision solution is highly specific to the flying game and sometimes the game may feel too heavy or light, depending on the environment and how the user is trying to fly. Children liked the prototype in the tests and often wanted to play it through several times.

In addition to the technical solution, this thesis describes the concept and the interaction design on a more general level. The author believes that the concept has several new ideas, since there are only a few earlier experiments of perceptually interactive computer games for children. The thesis also describes interesting observations from the usability tests, for example, relating to the spatial design of perceptual user interfaces.

The author was also responsible for the sound design of the prototype, although it was only a minor task compared to the interaction design. The author is quite satisfied with the results, consisting of two songs, one musical ambient sound track and the sound effects of the flying game.

Several novel applications of perceptual or other camera-based user interfaces will probably appear in the future, as desktop cameras become more common and their possibilities in user interface design are fully realized. In QuiQui's Giant Bounce, the camera was originally intended only for playing, but we also noticed that it could enable creating user profiles without the need of typing in one's name. The camera view of the user as a part of the screen graphics also makes it natural to explain things with examples, using prerecorded sequences of other users. QuiQui's Giant Bounce has also inspired other camera-based software, including the author's current project Animaatiokone (*http://animaatiokone.net*). Animaatiokone is an installation based on a simple tool for capturing and editing stop-motion animation, originally created for animating the characters of QuiQui's Giant Bounce.

Bibliography

[1] *http://www.vividgroup.com, 4th May 2002.*

[2] *http://www.gamesdomain.com/playstation2/reviews/ Jungle_Book_Groove_Party.html, 18th September 2002.*

[3] *http://www.stakes.fi/palvelut/palvelujen_laatu/lapset/kirjamessusadut/ sadut.asp, 17th September 2002.*

[4] *http://www.tombraider.com, 5th September 2002.*

[5] *http://www.computer-music.com/articles/artcl_musical_techniques-01.htm, 6th March 2002.*

[6] *http://www.thereminworld.com/learn.asp, 18th September 2002.*

[7] *http://www.cs.sfu.ca/˜amulder/ personal/vmi/HMTT.pub.html, 1st Nov 2001.*

[8] *http://homepage.mac.com/davidrokeby/home.html, 14th November 2002.*

[9] *http://www.oelinger.de/maria/interact/videoplace1.htm, 8th October 2002.*

[10] *http://www.listserv.acm.org/archives/wa.cgi?A2=ind0205E&L=chi-web&D=0&m=10716&P=720, 4th September 2002.*

[11] *http://www.lieroextreme.com, 5th September 2002.*

[12] *http://www.interplay.com/games/product.asp?GameID=114, 5th September 2002.*

[13] *http://www.robots.ox.ac.uk/˜ab/dynamics.html, 10th October 2002.*

[14] Directx 8.1 programmer's reference, *http://msdn.microsoft.com/library, 6th March 2002.*

[15] Andrew Blake and Michael Isard. *Active Contours* (Springer-Verlag London Limited, 1998).

[16] G.R. Bradski, Computer Vision Face Tracking For Use in a Perceptual User Interface, *Intel Technology Journal* (1998).

[17] J.L. Crowley, J. Coutaz, and F. Bérard, Things That See, *Communications of the ACM* **43** (2000).

[18] H. D'Hooge, Game Design Principles for the IntelPlay Me2Cam Virtual Game System, *Intel Technology Journal* (2001).

[19] Allison Druin. Cooperative inquiry: Developing new technologies for children with children. In *Proceedings of CHI'99*, pp. 592–599, 1999.

[20] Allison Druin. Children as our technology design partners: The surprising and the not-so-surprising. In *Proceedings of International Workshop on Interaction Design and Children*, pp. 1–2, Eindhoven, The Netherlands, 2002.

[21] W. T. Freeman, D. Anderson, P. Beardsley, C. Dodge, H. Kage, K. Kyuma, Y. Miyake, M. Roth, K. Tanaka, C. Weissman, and W. Yearazunis, Computer Vision for Interactive Computer Graphics, *IEEE Computer Graphics and Applications* pp. 42–53 (1998).

[22] W. T. Freeman, P. A. Beardsley, H. Kage, K. Tanaka, K. Kyuma, and C. D. Weissman, Computer Vision for Computer Interaction, *SIGGRAPH Computer Graphics Newsletter* **33** May 1999.

[23] W. T. Freeman, K. Tanaka, J. Ohta, and K. Kyuma. Computer vision for computer games. In *Proceedings of IEEE 2nd Intl. Conf. on Automatic Face and Gesture Recognition*, 1996.

[24] W. G. Gardner. Head tracked 3-D audio using loudspeakers. In *Proceedings of Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE ASSP)*, 1997.

[25] E. Bruce Goldstein. *Sensation & Perception* (Brooks/Cole Publishing Company, 1999).

[26] L. Hanna, K. Risden, and K. Alexander, Guidelines for Usability Testing with Children, *Interactions* **4**, 9–14 (1997).

[27] Johanna Höysniemi and Perttu Hämäläinen. Using Peer Tutoring in Evaluating the Usability of a Physically Interactive Computer Game with Children. In *Proceedings of International Workshop on Interaction Design and Children*, pp. 1–2, Eindhoven, The Netherlands, 2002.

[28] Emmanuelle C. Ifeachor and Barrie W. Jervis. *Digital Signal Processing A Practical Approach* (Addison-Wesley, 1993).

[29] K. Inkpen, W. Ho-Ching, O. Kuederle, S. Scott, and G. Shoemaker. “This is fun! We’re all best friends and we’re all playing.“: Supporting children’s synchronous collaboration. In *Proceedings of Computer Supported Collaborative Learning (CSCL) ’99*, Stanford, C.A., 1999.

[30] Jean-Marc Jot, Real-time spatial processing of sounds for music, multimedia and interactive human-computer interfaces, *Multimedia Systems* **7**, 55–69 (1999).

[31] Ismo Karvinen. *Lapsi ja urheilu perustietoa liikunnasta ja urheilusta ohjaajille, opettajille ja lasten vanhemmille* (Otava, 1991).

[32] Erwin Kreyzig. *Advanced Engineering Mathematics, Seventh Edition* (John Wiley & Sons, Inc., 1993).

[33] M. Krueger, T. Gionfriddo, and K. Hinrichsen. VIDEOPLACE - An Artificial Reality. In *Proceedings of CHI 85*, 1985.

[34] Petri Kuljuntausta. *ON/OFF Eetteriäänistä sähkömusiikkiin* (Like kustannus ja Kiasma, 2002).

[35] T.B. Moeslund, Computer vision-based human motion capture - a survey, *Technical report: Aalborg University, Laboratory of Computer Vision and Media Technology* (1999).

[36] N. Oliver, A. P. Pentland, and F. Berard. LAFTER: lips and face real time tracker. In *Proceedings of Computer Vision and Pattern Recognition (CVPR’97)*, pp. 123–129, 1997.

[37] Stephen E. Palmer. *Vision Science: Photons to Phenomenology* (MIT Press, 1999).

[38] C.S. Pinhanez, A.D. Wilson, J.W. Davis, A.F. Bobick, S. Intille, B. Blumberg, and M.P. Johnson, Physically Interactive Story Environments, *IBM Systems Journal* **39**.

[39] K. Sengupta, H. Wong, and P. Kumar. Computer vision games using a cheap (<100$) webcam. In *Proceedings of 6th International Conference on Control, Automation, Robotics and Vision (ICARCV’2000)*, Singapore, 2000.

[40] A.K. Sinha and J.A. Landay. Visually prototyping perceptual user interfaces through multimodal storyboarding. In *Proceedings of Workshop on Perceptual User Interfaces*, Orlando, Florida, 2001.

[41] M. Sonka, V. Hlavac, and R. Boyle, editors. *Image Processing, Analysis and Machine Vision, 2nd edition* (Brooks/Cole Publishing Company, 1999).

[42] J. Stewart, B.B. Bederson, and A. Druin. Single display groupware: A model for co-present collaboration. In *Proceedings of CHI 99*, pp. 286–293, Orlando, Florida.

[43] K. Subrahmanyam, R.E. Kraut, P.M. Greenfield, and E.F. Gross, The Impact of Home Computer Use on Children's Activities and Development, *Children and computer technology* **10** (2000).

[44] Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers. Wallflower: Principles and Practice of Background Maintenance. In *Proceedings of the Internal Conference on Computer Vision*, Corfu, Greece, 1999.

[45] C.R. Wren, F. Azarbayejani, A. Darrel, T. Davis, J. Starner, A. Kotani, C. Chao, M. Hlavac, K. Russel, A. Bobick, and A. Pentland. Perceptive spaces for performance and entertainment (revised). In *Proceedings of ATR Workshop on Virtual Environments*, Kyoto, Japan, 1998.

A Software architecture of the prototype

Computer games are comprised of several software systems. In addition to the actual game behavior, the systems implement other features, such as drawing the graphics or playing the sounds.

Usually it is not practical to program everything yourself. There are several commercial libraries available, including graphics and sound engines as well as whole development environments. Easy to use and versatile tools, such as Macromedia Flash and Director, have not traditionally offered enough performance for commercial game development, as opposed to the engines used in games, such as Quake and Max Payne. However, this is changing as the current version of Director already features hardware accelerated three dimensional graphics.

The software tools used for QuiQui had the following requirements:

- Good quality end product in a short time and at low cost.
- Free distribution of the game.
- The tools should support collaboration between programmers, sound designers and visual designers. In practice this means that people can make changes to files without affecting the work of others. The tools should also minimize the need for converting files from one format to another.
- Efficient implementation of the video and audio processing.
- Small size of the game for delivery via the Internet.
- Use of two-dimensional cartoon graphics.

The architectural solution of the prototype is presented in Figure A.1. The graphics were designed in Macromedia Flash 5 and the Flash 5 player was used as a graphics engine. Flash is an efficient production environment for 2-D graphics and there was no need to convert the image files to other formats. The player is available free of charge from Macromedia's website.

The game was not programmed in Flash because of the lack of a proper debugger with breakpoints. The author was used to using breakpoints and debugging ‘blindly’ was cumbersome in Ujo Sankari that was programmed in Flash. The game was programmed in Borland Delphi 4 using Delphi object Pascal. Delphi allows a Flash player to be easily integrated with the program as an ActiveX component. Using Delphi also enabled more fluent collaboration between the programmer (the author) and the visual artists (Teppo Rouvi and Johanna Höysniemi). Because the Flash movie clips did not contain any scripting, they could be modified and replaced by the graphics designer. The game program only requires that the Flash movie contains movie clip instances with certain names. The game accesses the movie clips by their names to change their coordinates, visibility, frame number etc.

The sound engine and perceptual computing, that is, the computer vision and hearing were programmed using Microsoft Visual C++. The C++ language is better suited for mathematical purposes and Delphi has no official support for the Microsoft DirectX8 API needed for accessing the camera and sound card drivers.

Data flows according to the arrows in Figure A.1. The raw video and audio data is analyzed to provide the control signals for the game logic. The game logic controls the sound engine and the Flash player.

In principle, the sounds could be played using Flash, but the sound output functionality of Flash is rather limited. For example, Flash does not provide means for altering the frequency of the played sound in realtime, an effect used in the hairdryer sound of the windmill. The custom built sound engine used the DirectX 8 API for sound output and the free Ogg Vorbis audio compression technology from Xiphophorus. Ogg Vorbis enables the compression of audio much like the popular MPEG-1 layer 3 (mp3) technology, but the software libraries needed are available for free.

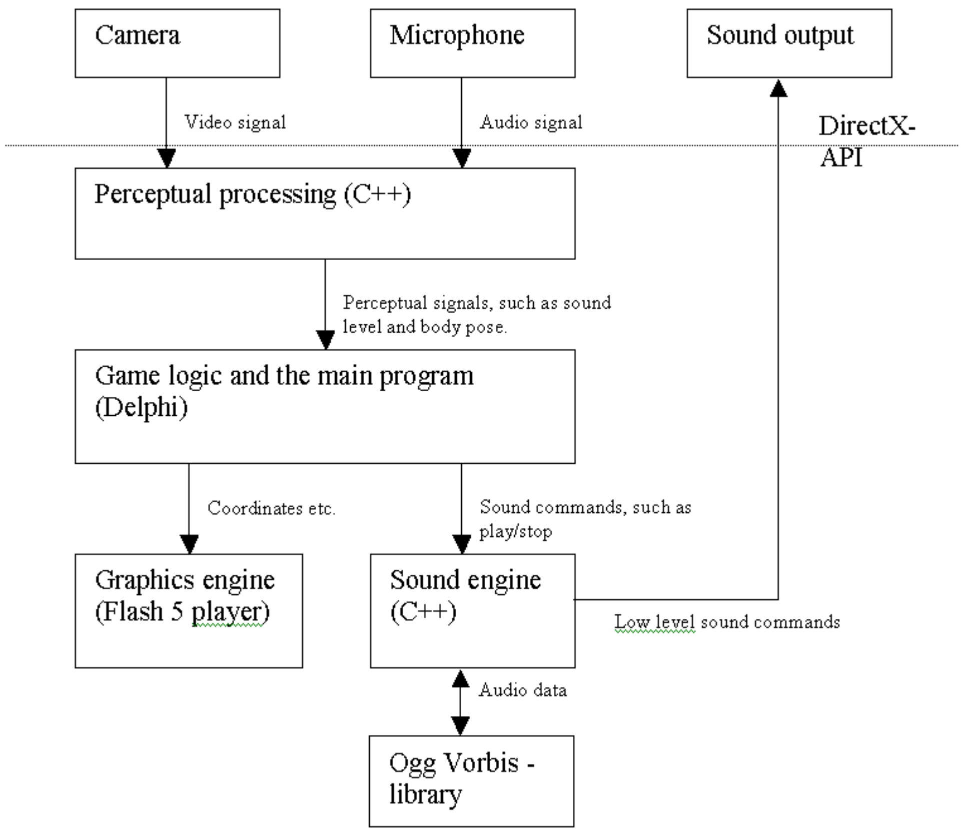
Camera
Microphone
Sound output
Video signal
Audio signal
DirectX-API
Perceptual processing (C++)
Perceptual signals, such as sound level and body pose.
Game logic and the main program (Delphi)
Coordinates etc.
Sound commands, such as play/stop
Graphics engine (Flash 5 player)
Sound engine (C++)
Low level sound commands
Audio data
Ogg Vorbis - library

Figure A.1: A block diagram of the software architecture

B Computer vision source code

This section contains the C++ source code of the relevant methods of the CFlyingSensor class that implements the computer vision methods described in this thesis. The code is included to give a concrete example of what was done for those not so familiar with programming and computer vision. Note that the source code contains calls to image processing functions not printed here, but the meaning of the calls is explained in the comments. The total size of QuiQui's Giant Bounce is about 6000 lines of code and it would be impractical to include all the code in this thesis.

```
#include "math.h"
#include "flyingsensor.h"
#include "debugprintf.h"
#define ALGORITHM2
const int PROCESSWIDTH=120;
const int PROCESSHEIGHT=96;

/**

Test whether the input image (pointed to by pBuff) contains a flying user or not.
The input image is assumed to be a thresholded temporal difference image.

*/

bool CFlyingSensor::paraboloidCriteria(BYTE *pBuff, int width, int height, bool visualize){
    int i;
    float centerx;
    float topx[1000],topy[1000];

    //get the topmost non-zero pixels
    int npixels=getTopPixels1C(pBuff,width,height-2,topx,topy,&centerx);
    int topwidth=topx[npixels-1]-topx[0];

    //plot the topmost pixels if needed
    if (visualize){
        for (i=0; i<npixels; i++){
            putPixel1C(pBuff,width,height, topx[i],topy[i]);
        }
    }

    //prepare for curve fitting
    float sig[1000];
    for (i=0; i<npixels; i++)
        sig[i]=10;
    static float chisq,aa[6];
    float olda3=aa[3];
    int ia[6];
    ia[1]=ia[2]=ia[3]=ia[4]=ia[5]=1;
    //fit a paraboloid to the tompost pixels
    lfit(topx,topy,sig,npixels-1,aa,ia,3,&chisq,paraboloid); //fit a paraboloid

    //if the quadratic coefficient aa[3] is negative enough and the x coordinate of the peak
    //of the paraboloid is near to the center of the topmost pixels, the image probably
    //contains a standing or walking user
    if ((aa[3]<-0.05) && (fabs(-aa[2]/(2*aa[3])-centerx)<topwidth/4)){
        DebugPrintf("x^2 factor %f\n",(double)aa[3]);
        plotPoly(pBuff,width,height,20,2,&aa[1],8);
        return false;
    }

    //compute the direction of the thrust force from the vertical movement of the edge pixels
    float yy=0;
    for (i=0; i<npixels; i++)
        yy+=topy[i];
    yy/=npixels;
    float ydiff=yy-cy;
    cy=yy;
    ydir=sign(ydiff);
    return true;
}

const double pi=3.141592653589793;


/**

Analyze the 24 bit RGB input image (pointed by pBuff) and update the following member variables so
that they can be queried by the host program:

    force                       the "thrust" force
    leftAngle, rightAngle       the angles of the left and right arm

*/

void CFlyingSensor::processFrame(BYTE *pBuff,int width, int height, int deltaTime, bool visualize){
    static int myDeltaTime=0;
    //visualize=true;
    //convert to black and white and smooth
    conv3C1C(pBuff,pBuff,width,height);

    //The sensoring algorithm does not actually need too high a sample rate, it can even lead
    //poor performance since there may be not enough difference pixels. We simply drop frames
    //according to the deltatime
    deltaTime+=myDeltaTime;
    if (deltaTime<1000/16){
        conv1C3C(pBuff,pBuff,width,height);
        myDeltaTime=deltaTime;
        return;
    }
    myDeltaTime=0;

    //copy to a temporary buffer and lowpass filter using a rectangular kernel
    //(spatial averaging)
    memcpy(temptemp,pBuff,width*height);
    boxFilter1C(temptemp,width,height,8,3);

    //downsample to reduce the amount of data to process
    BYTE *origBuff=pBuff;
    int origWidth=width,origHeight=height;
    width=PROCESSWIDTH;
    height=PROCESSHEIGHT;

    //processBuff and prevFrame are instances of a dynamic buffer class. they reallocate the
```

```
//buffer if its width and height change. We never know what image sizes the Windows
//capture drivers output.
processBuff.update(width,height);
prevFrame.update(width,height);

//stretchBlt1C(processBuff.buff,width,height,tempBuff.buff,origWidth,origHeight);
//stretchBlt1C scales the image
stretchBlt1C(processBuff.buff,width,height,temptemp,origWidth,origHeight);
pBuff=processBuff.buff;

//compute the absolute value of the difference to previous frame
absDiffToPrev1C(pBuff,prevFrame.buff,width,height);

//compute a histogram
int histogram[256];
hist1C(pBuff,histogram,width,height);

//find the first local minima of the histogram after the value
//corresponding to half of the amount of pixels (the basic assumption is
//that at least half of the difference image contains noise and no real
//moving objects)
int total=0;
static int thresh=127;
for (int i=0; i<256; i++){
    total+=histogram[i];
    if (total>=width*height/2)
        break;
}
i++;
for (; i<254; i++){
    total+=histogram[i];
    //if a local minima is found, update the threshold
    if (histogram[i]<=histogram[i-1] &&
        histogram[i]<histogram[i+1]){
        thresh=i;
        break;
    }
}

//Subtract the found value to reduce noise before any features are extracted
//note that the function clips negative values to zero
subConst1C(pBuff,thresh,width,height);

/*
//Compute mean and variance of the difference values from the histogram
double mean=0;
for (i=0; i<256; i++){
    mean+=histogram[i];
}
mean/=width*height;
double variance=0;
for (i=0; i<256; i++){
    variance+=histogram[i]*(mean-i)*(mean-i);
}
variance/=width*height;
double stdev=sqrt(variance);

//Extract features only if standard deviation is large enough (not only noise)
if (stdev>3){*/

//Extract features only if enough difference pixels found
if (width*height-total>width/2){
    //check whether the user is flying or not
    if (paraboloidCriteria(pBuff,width,height,visualize)){
        //Assuming that the user is flying, compute two spatially biased mass
        //centra so that the moving hands of the user attract them
```

```
ULONG   leftx=0,lefty=0,rightx=0,righty=0,
        leftTot=0,rightTot=0,xx=0,yy=0,tot=0;
for (int y=0; y<height; y++){
    BYTE *p=pBuff+y*width;
    for (int x=0; x<width; x++){
        int val=p[x];
        if (val){
            tot++;

            //val*=val;
            int w=(val*(x*x+height-y)) >> 4;
            leftx+=x*w;
            lefty+=y*w;
            leftTot+=w;

            w=(val*((width-x)*(width-x)+height-y)) >> 4;
            rightx+=x*w;
            righty+=y*w;
            rightTot+=w;
            xx+=x*val;
            yy+=y*val;
        }
    }
}

//measure the height of the area containing most of the difference
//the square of this is used as a measure against which the amount of
//change is compared
int size=width*height;
int tot2=0;
int thr=tot/40;
for (i=0; i<size; i++){
    if (pBuff[i]){
        tot2++;
        if (tot2>thr)
            break;
    }
}
int y1=i/width;
tot2=0;
for (i=size-1; i>=0; i--){
    if (pBuff[i]){
        tot2++;
        if (tot2>thr)
            break;
    }
}
int y2=i/width;
float relArea=(y2-y1)*(y2-y1)/2;
if (visualize){
//  drawLine1C(pBuff,width,height,0,y1,width-1,y1);
//  drawLine1C(pBuff,width,height,0,y2,width-1,y2);
}

//the slope of the line connecting the two centra determines
//user rotation
float oldcy=leftcy+rightcy;
if (leftTot>0 && rightTot>0){
    leftx/=leftTot;
    lefty/=leftTot;
    rightx/=rightTot;
    righty/=rightTot;
    float c1=0.5,c2=0.5;
    leftcx=c1*leftcx+c2*leftx;
    leftcy=c1*leftcy+c2*lefty;
    rightcx=c1*rightcx+c2*rightx;
```

```
                rightcy=c1*rightcy+c2*righty;

                headAngle=-atan((leftcy-rightcy)/(rightcx-leftcx))*1.1;//1.25;
                headAngle=min(pi/6,max(-pi/6,headAngle));
            }

            //compute the vertical direction of the movement
            ydir=sign(leftcy+rightcy-oldcy);

            //compute thrust force
            float force;
            force=min(relArea,tot)/(relArea); //normalized to 1
            force=force*deltaTime/85;  //added after testing with different cameras
            if (ydir>0)
                force=force*0.3;    //force is zero if wings move upwards
            //DebugPrintf("angle %f,   force %f\n",(double)headAngle,(double)force);

            float temp=force*(1.0+cos(headAngle));
            if (temp>rightForce)
                rightForce=temp;
            temp=force*(1.0-cos(headAngle));
            if (temp>leftForce)
                leftForce=temp;

            //compute wing angles, incrementing or decrementing them depending on
            //the direction of the movement
            leftAngle=leftAngle+ydir*3.1415/4;
            if (leftAngle>3.1415/4)
                leftAngle=3.1415/4;
            if (leftAngle<-3.1415/4)
                leftAngle=-3.1415/4;

            //in this version, we set the left and right angles to be equal
            rightAngle=leftAngle;
        }
    }
    //done processing, visualize
    if (visualize){
        drawLine1C(pBuff,width,height,
                leftcx-(rightcx-leftcx),
                leftcy-(rightcy-leftcy),
                rightcx+(rightcx-leftcx),
                rightcy+(rightcy-leftcy));
        drawLine1C(pBuff,width,height,leftcx,leftcy-5,leftcx,leftcy+5);
        drawLine1C(pBuff,width,height,rightcx,rightcy-5,rightcx,rightcy+5);

        //scale back to the original size
        stretchBlt1C(origBuff,origWidth,origHeight,processBuff.buff,width,height);
    }

    //convert the processed image back to 24 bit RGB for visualizing
    conv1C3C(origBuff,origBuff,origWidth,origHeight);
}
```

C The synopsis behind the prototype

The following pages contain the synopsis used when designing the prototype. It is written in Finnish by Laura Turkki. Included here in its original form, it was originally written only for internal use of the project.