RWTH AACHEN UNIVERSITY

PinguTouch

Investigating Multi-Touch Technology for Collaborative Casual Gaming

Diploma Thesis at the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Christian Mattar

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Dr. T.C. Nicholas Graham

Registration date: Jul 11th, 2007
Submission date: Dec 19th, 2007

I hereby affirm that I composed this work independently and used no other than the specified sources and tools and that I marked all quotes as such.

Hiermit versichere ich, diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht zu haben.

Aachen, 17.12.2007

(Christian Mattar)

Contents

List of Figures

List of Tables

Abstract

Multi-touch tabletop gaming allows the social experience that a classical board game offers to be integrated with the interactive capabilities of modern computers. While several existing tabletop systems exploit these aspects to some degree individually, none of them makes full use of both the social and the interactive possibilities.

PinguTouch is a casual cooperative game created to fill this gap, and tap the full potential of the properties of a multi-touch tabletop setup. The game involves players performing hand gestures on the table surface to aid penguin-like creatures, called pingus, through an obstacle course. Three gestures are implemented: flicking of pingus, pulling of levers, and making bridges or blockades.

To fully utilize the co-located scenario, PinguTouch encourages close spatial cooperation by employing quasi-modes. Since the number of actions one player can perform at any one point of time are restricted, players must work closely together in a common workspace to succeed in rescuing a large number of pingus. This enables a game environment impossible to recreate in a non co-located scenario.

To further support interpersonal interaction similar to a typical board game, the visual layout of the game is fashioned in an octagonal form. The “down”-direction of the game world leads towards the radius of the octagon, while the “up”-direction leads towards its center, thus allowing all users standing around the table easy access to their part of the game world.

The system was set up in the Industrion museum, Kerkrade, Netherlands. After including several improvements based on user observations, a systematic evaluation was performed involving typical museum visitors. Detailed video analysis and survey results show that PinguTouch is a highly cooperative and enjoyable game.

Überblick

Der Einsatz eines multitouch-fähigen Spieltisches ermöglicht die Integration des sozialen Erlebnisses eines klassischen Brettspieles mit den interaktiven Möglichkeiten moderner Computer. Obwohl einige bereits vorhandene interaktive Spieltische diese Aspekte einzeln zu einem gewissen Grad ausnutzen, schöpft keines von ihnen sowohl die sozialen als auch interaktiven Möglichkeiten voll aus.

PinguTouch ist ein kooperatives Gelegenheitsspiel, das entwickelt wurde, um diese Lücke zu füllen und das gesamte Potenzial eines multitouch-fähigen Tisches zu nutzen. Das Spiel erlaubt es den Benutzern, Handgesten auf der Tischoberfläche auszuführen. Dadurch können Pinguin-artige Wesen, die Pingus, über Hindernissen hinweg geholfen werden. Drei Gesten wurden implementiert: das Anschubsen von Pingus, das Ziehen an Hebeln und das Platzieren von Brücken oder Blockaden.

Um eine möglichst gesellige Situation um den Spieltisch zu erzeugen, sind alle diese Handgesten quasi-modal. Da so die Anzahl Aktionen, die ein einzelner Benutzer gleichzeitig ausführen kann, beschränkt ist, müssen die Spieler in einem gemeinsamen Arbeitsbereich eng zusammenarbeiten, um alle Pingus retten zu können. Auf diese Weise entsteht eine Spiel-Umgebung, die in einer Situation ohne gemeinsam genutztes Spielfeld nicht möglich wäre.

Um die typischen zwischenmenschlichen Interaktionen eines Brettspieles zu ermöglichen, ist das Spielfeld in einer oktagonalen Form angeordnet. Die Richtung "unten" der Spielwelt führt in Richtung des Radius des Oktagons, während "oben" in Richtung des Mittelpunkts zeigt. Dadurch erlangen alle Nutzer des Tisches einfachen Zugriff auf den vor ihnen liegenden Teil der Spielwelt.

Das System wurde im Industrion Museum in Kerkrade, Niederlande, installiert. Nachdem basierend auf Benutzerbeobachtungen einige Verbesserungen vorgenommen wurden, wurde an Hand typischer Museumsbesuchern eine systematische Benutzerstudie durchgeführt. Eine detaillierte Video-Analyse zusammen mit Umfrageergebnissen haben gezeigt, dass PinguTouch ein unterhaltsames, hoch-kooperatives Spiel ist.

Acknowledgements

The PinguTouch project would not have been possible without the support of many people, to all of whom I owe my gratitude.

First of all, I thank Prof. Jan Borchers for sparking my interest in HCI. He enabled me to create PinguTouch and always gave me the resources I asked for. I also thank Dr. Nick Graham from Queen's University for agreeing to be second examiner.

It was David Holman who introduced me to multi-touch technology and supervised the implementation part of this thesis. Eric Lee continued where David left off, supervised the evaluation part, and performed countless correction readings. Without their guidance, PinguTouch and this thesis would have been of considerably worse quality.

Elaine Huang, Marius Wolf, and Jonathan Diehl shared their experience with me regarding user studies. In a similar way, Jonas Lang helped me increase my understanding of statistics. The evaluations would have been much more difficult without their valuable advice.

Wouter van Dillen and the crew from the Industrion museum patiently supported me during my countless evaluation trials and we have developed an excellent work relationship over these months.

I thank all the authors of the non-commercial software that I used: Ingo Ruhnke and the contributors to Pingus—building this game from scratch would have been

impossible; David Wallin and the contributors to TouchLib; Michael Kipp for creating Anvil and allowing me to use it to analyze the video data.

On a more personal note, I thank my study-buddies Georg, Rene, Seppel, Andre, and Tobias for five years of geeky fun, work, and friendship. I never could have made it without you! Here's to another five years!

Finally, and most importantly, I want to thank my parents, Carla and Ernst Mattar, for making my studies possible. You allowed me to keep my focus on computer science without having to think too much about how to pay next month's rent.

Thank you!

Conventions

Throughout this thesis we use the following conventions:

Source code and implementation symbols will be typeset using a `monospace font`.

The plural "we" will be used throughout this thesis instead of the singular "I", even when referring to work that was primarily or solely done by the author.

Unidentified third persons are always described in male form. This is only done for purposes of readability.

The whole thesis is written in American English.

Chapter 1

Introduction

> *"Most people think video games are all about a child staring at a TV with a joystick in his hands. I don't. They should belong to the entire family. I want families to play video games together."*
>
> *—Shigeru Miyamoto, designer of Mario, in a BusinessWeek interview*

Games have always been part of civilization and pervade all of human history. They are a focus of social gatherings, and can be used for practicing physical fitness, social rules and skills, or intellect. From simple Roman dice to current board games, many different forms exist.

Games have always existed in many forms.

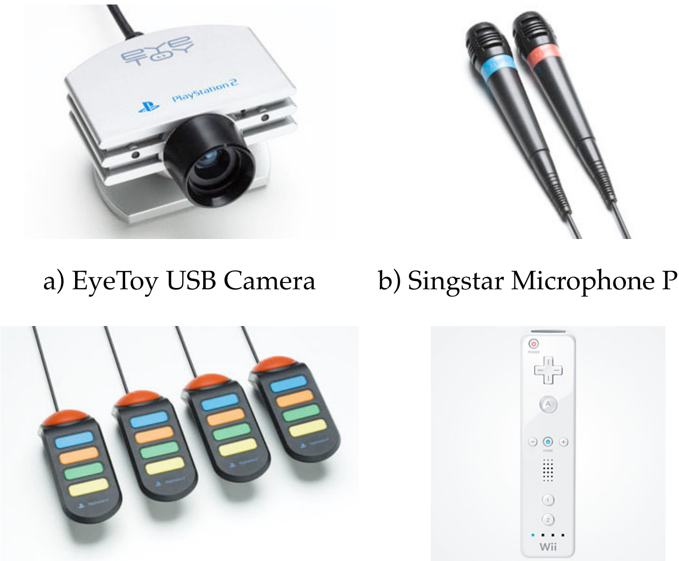
With the release of Pong in 1972, the era of video games was heralded in. The industry is booming: since 2001, it has generated more revenue per year than the U.S. box office.[1] Traditionally, video gaming consoles have mainly been catering to the male demographics of below age 35.[2] However, in the last few years, it has been shown that with titles like EyeToy, Singstar, Buzz, and most recently, Wii Sports, a more general demographic can be enticed into

Specialized input devices can make video games more accessible.

[1] Compare http://www.oecd.org/dataoecd/19/5/34884414.pdf to http://www.clickz.com/973421.

[2] See http://web.archive.org/web/20030801073725/www.theesa.com/EF2003.pdf.

computer gaming.[3] All of these games have game mechanics that can be easily learned and utilize input devices that give continuous feedback, with functions that are immediately recognized by the users (see Figure 1.1). Even though the Nintendo Wii Remote is a more generic device compared to the others, it still allows to perform movements modeled on real-world interactions.



a) EyeToy USB Camera

b) Singstar Microphone Pack

c) Buzz! Buzzers

d)Wii Remote

Figure 1.1: Various game input devices; **a)-c)** taken from `http://uk.playstation.com/ps2/hardware/accessories/`; **d)** taken from `http://www.nintendo.com/accessorieswii`.

The often-used gamepad on the other hand (see Figure 1.2), while quite efficient for a wide range of games, builds a strong abstraction layer between the intent of the user and the outcome. It is "just button pressing", whereas specialized input devices not only increase efficiency and satisfaction, but also immerse the player much more into the application [Borchers, 2001].

Multi-touch hugely increases bandwidth upon single-touch.

In terms of general-purpose input devices, touch input is a step in that same direction, since input and output are

[3] See minutes 07:00 to 12:00 in webcast at

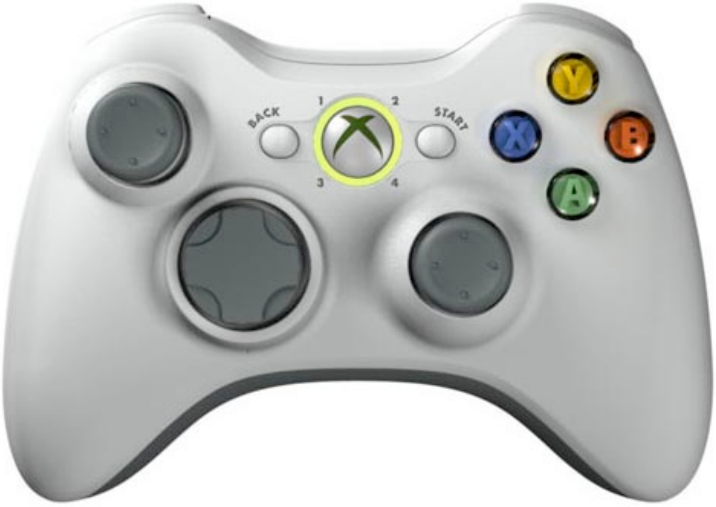


Figure 1.2: The Microsoft Xbox 360 Wireless Controller, an example of a typical gamepad; taken from `http://www.xbox.com/en-US/hardware/x/xbox360wirelesscontroller/default.htm`.

coupled closely from a spatial perspective. The success of the Nintendo DS portable gaming system (see Figure 1.3), which advertises single-touch input as one of its main features, shows the merits of this input mechanism. Until now, however, the very low bandwidth of single-touch with only one point in a two-dimensional grid made this method inappropriate for multi-player gaming situations.

Multi-touch input alleviates this concern: the ability of recognizing multiple points of touching at once hugely increases the input bandwidth. Multi-user applications now become possible and interaction with such devices should be much more natural, since they cannot be easily overloaded.

Multi-touch enables whole-hand gestures.

Another advantage of multi-touch is the ability to recognize whole areas of contact. Users can not only touch the surface with single fingers, but can use the whole hand, thereby performing input gestures that are actually relevant to their underlying semantics (cf. Section 2.3—"Tabletop applica-

http://ms.nintendo-europe.com/e32007/enGB/media.html.

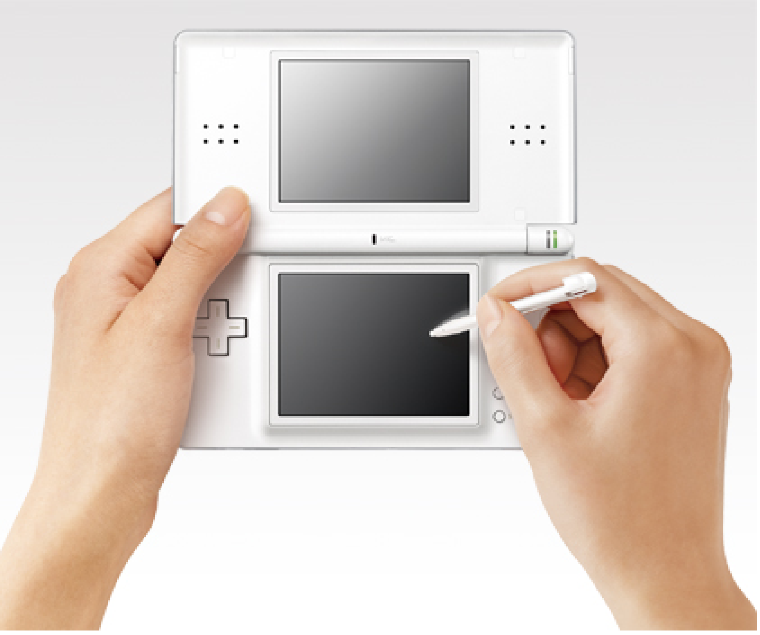


Figure 1.3: The Nintendo DS Lite system; taken from `http://www.nintendo.co.jp/ds/lite/gallery/image7.html`.

Precision and ergonomics are weak points.

tions involving complex hand gestures"). Some of the disadvantages compared to other input devices, e.g., the mouse, remain: precision is dependent on the thickness of the user's finger and the constant movement of the user's arms leads to bad ergonomics for long usage spans.

Tabletop setups improve the social experience.

To further increase appeal to casual users, a horizontal table setup can be used to mimic a traditional board game. As observed by Magerkurth et al. [2004], board games offer a unique experience. Players not only interact with the game board, but also with each other. In contrast, during usual computer gaming interaction, the players sit in a row next to each other, and the attention is completely focussed on the output device, i.e., the screen. Even when users make a conscious effort to shift their attention to the other players, recognizing the other players' actions by observing their handling of the input device is barely possible.

While augmenting traditional tabletop experiences with computer technology is being heavily investigated

Multi-touch enables real-time multiplayer gaming on a tabletop.

[Magerkurth et al., 2005], pure computer tabletop gaming has had little exposure. One reason might be that with single-touch technology, such a system would either be a single-player experience or a turn taking game. An interactive single-player game is usually much better suited for a gaming console or a PC, since the social component does not have to be considered. Turn taking games are better served by including real game pieces, since the interaction with physical objects is much more immediate and real-time feedback about state changes of the game world is not as important. With the advent of low-cost multi-touch technology, however, development of real-time multiplayer tabletop gaming becomes feasable.

PinguTouch uses the unique features of a multi-touch tabletop.

In this thesis, we present the development process of "PinguTouch", a collaborative casual game using the unique features of a multi-touch tabletop system. It was our goal to demonstrate a wide range of possibilities this new input method offers over previous ways of gaming:

- A highly real-time scenario that would have been impossible to implement when including real game pieces.
- Full exploitation of the co-located tabletop design regarding the social experience, making the game concept hard to recreate involving other setups.
- Whole hand gestures that go beyond actions typically associated with general purpose input devices.

PinguTouch was installed at a dedicated multi-touch table at the Industrion-museum in Kerkrade, Netherlands,[4] as part of their "Games, let's play!"-exhibition (see Figure 1.4).

1.1 Cooperative gaming

To make full use of the co-located tabletop setup, we decided that a cooperative gaming experience would be a

[4] See http://www.industrion.nl.

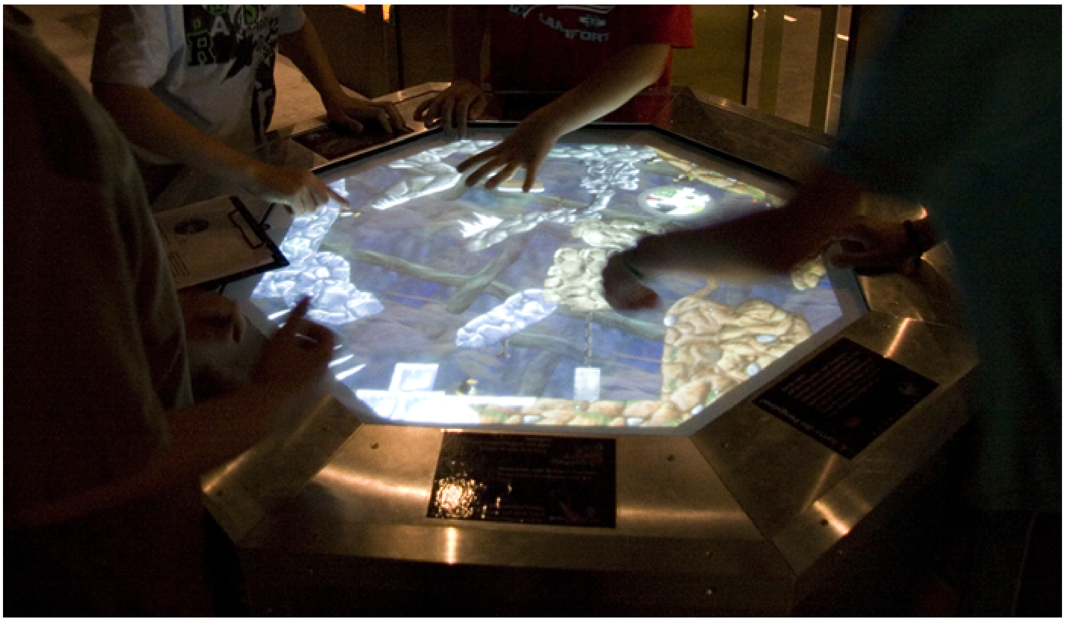


Figure 1.4: PinguTouch installation at the Industrion museum.

new, interesting way to encourage interaction between the users.

Conflict is part of most games.

Many traditional board games are competitive in nature. While several titles include opportunity for building teams, only few allow for all players to play on the same side. The reason for the lack of games of this kind lies in the nature of gaming: conflict is an important part of any game as defined by Crawford [1984]. As there is little possibility for other active elements in non-electronic games, the conflict either has to unfold between the human players, or random elements have to be added that control the adversarial side.

Random events can control adversaries, but makes winning dependent on chance.

A good example for a well-received purely cooperative title is "Lord of the Rings", distributed by Fantasy Flight Games. Players have to pool their gained cards and tokens to be able to progress through the game (see Figure 1.5). The adversary is controlled by dice rolls and chance cards. This randomness is often described as the greatest weakness of the game: although strategy plays a large part in success, an element of chance is always necessary to keep the game challenging even for advanced players. With the highly interactive experience a multi-touch screen can offer, the computer can take the part of the adversary. Cooperation becomes more feasible and an interesting avenue to explore.

Figure 1.5: The cooperative Lord of the Rings board game; taken from `http://www.fantasyflightgames.com`; reprinted with permission.

Cooperation requires interdependence of outcome and means.

Cooperation is usually based on joint discussion about strategy and the concerted execution of that strategy. A tabletop surface lends itself well to that kind of activity, as observed in daily situations like business meetings or study groups. However, not every game is suitable for a cooperative play style. To actually promote cooperation among players, two conditions must hold as observed by Johnson and Johnson [1992]:

- **Outcome interdependence between players** is necessary for the players to actually have an interest in cooperation.
- **Means interdependence between players** is necessary for the players to have the opportunity of cooperation.

Outcome interdependence is easily achieved in a game, since goals are usually abstract and can be arbitrarily set. The simplest way to accomplish this is by setting the exact identical goal for all players, so that either everyone wins or everyone loses.

Physical space is an important part of tabletop interaction.

Means interdependence needs to be examined in more detail. One important new resource offered by a multi-touch table over a desktop system is space. While on a desktop all areas are well reachable thanks to mouse acceleration and similar techniques, the physical length of a tabletop and comparingly slow and strenuous movement of the arms can restrict a player. Natural cooperation can emerge by simply partitioning the table into zones of responsibility: one player handles one half of the table, while the other player handles the other half. While this approach is valid, it only takes advantage of the co-located nature of tabletop gaming during discussion phases. During execution, interaction between the players is restricted to the edges of their respective zones.

Quasi-modes encourage cooperation.

Another interdependent means has to be added. Since interaction on a multi-touch table takes place by hand gestures, the usage of quasi-modes [Raskin, 2000] suggests itself. If players can only assert control over the game world in two places at the same time, they are encouraged to be more flexible in choosing their space. Players acting alone can easily be overwhelmed and need the assistance of others to ensure a good result for the whole team.

PinguTouch encourages concurrent use of workspace.

PinguTouch is specifically designed to create a new game experience by emphasizing concurrent usage of workspace instead of neatly dividing the table. Although under everyday conditions, people usually keep their distance depending on familiarity among each other, the success of titles like "Twister"[5] suggests that while engaged in gaming, users quite enjoy the unusual situation.

1.2 Peculiarities of game design

Task knowledge is meaningless when designing games.

For the development of any task-based application, the designer should have a very good idea about the nature of its users and the nature of the tasks the users perform. In the context of designing a game, however, "task knowledge" is a meaningless term. The user wants to be entertained,

[5] See http://www.hasbro.com/default.cfm?page=browse&brand=667.

the particular mechanics and goals involved are up to the developer.

Game mechanics and user interface greatly influence each other.

Furthermore, the line between user interface and game mechanics is often blurred. The game “Monkey Ball” by Amusement Vision, first released in 2001 as an arcade game, serves as a graphic example. Its goal is simple: lead a monkey trapped in a ball through an obstacle course in less than 60 seconds while avoiding to fall into pits.

A very efficient way of accomplishing that goal could be controlling the game from a top-down view. The player could determine the route the monkey should take by setting waypoints with his mouse, and upon clicking a start button, the monkey would start moving.

In reality, the game presents the player with a third person perspective of the monkey (see Figure 1.6). It can be directly controlled by an analog thumbstick and a lot of experience is needed to be able to perform the necessary motions with the right intensity. Quite clearly, this less efficient interface leads to a decidingly different and probably more fun game.

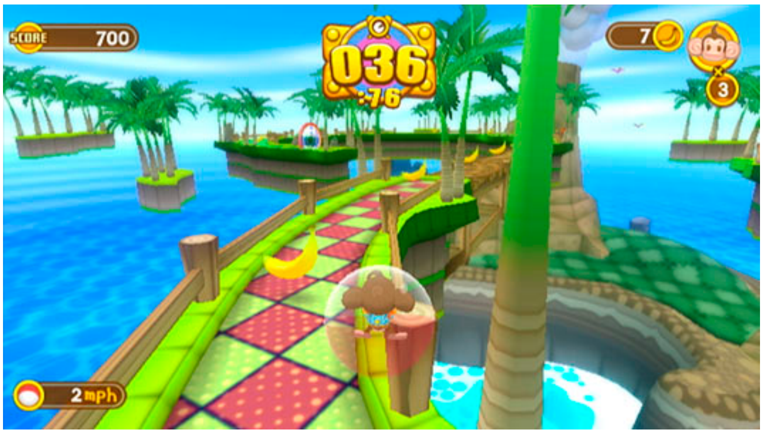


Figure 1.6: A screenshot of Super Monkey Ball: Banana Blitz; taken from `http://www.sega.com`.

Game mechanics have to be adjusted with the physical interface.

This example also demonstrates how the game mechanics have to be designed with the possibilities and restrictions of the physical interface in mind. Without an input device similar to a joystick available, this type of game would not be possible. We expect that it will take several approaches until the interaction metaphors most suitable to multi-touch gaming will be found. This thesis can only be a first step towards that.

1.3 Thesis structure

The thesis is organized as follows:

- In *Related work*, we describe research on real-world tabletop interactions, multi-touch technology, and several task-based and gaming-related tabletop applications.
- In *Designing the game*, design decisions for PinguTouch and their reasoning are presented.
- In *Refining the game*, problems identified during user tests are described and their solutions presented.
- *Implementation details* describes the code base of the game.
- *Evaluation* describes user observations designed to test whether we accomplished our design goal of creating an entertaining cooperative multi-touch game.
- In *Summary and future work*, we summarize our conclusions and present ideas for further improvement of the work.

Chapter 2

Related work

"It usually starts with the research.
I'll find some subject that I'm reading about
that fascinates me. It will pique my interest
and then I'll slowly become obsessed with it."

—Will Wright, designer of "The Sims",
in an interview with GIGnews.com, April 2002

In this chapter, we will discuss several previous research efforts that this work touches upon.

Since creation of a social situation similar to a board game lies at the core of PinguTouch, studies of human behavior around tabletops will be described first. After that, the related hardware and technology we used to enable such situations will be explained. Then, we will describe tabletop applications making heavy use of complex hand gestural interaction. Finally, tabletop games allowing for full cooperation between all players will be described.

2.1 Social behavior in tabletop environments

In this section, we will review two previous studies that analyze behavior in cooperative tabletop situations.

2.1.1 Territoriality in collaborative tabletop workspaces

The work by Scott et al. [2004] examines people's behavior regarding space on a traditional non-interactive desk tabletop.

Two studies were performed: in the first one, users were observed when performing entertainment tasks like solving puzzle games, playing with Lego, or playing the Pictionary game. In the second one, groups of two or three were observed creating a furniture layout for a reading room in a library from paper and cardboard.

The tabletop space gets divided into personal, group, and storage areas.

According to the findings, each user partitioned the table into three territories: personal, group, and storage (see Figure 2.1). Personal territories were the areas on the edge of the table directly in front of each user. 87% - 100% of the actions in that area were performed by the user next to it. This partitioning happened without any explicit negotiation with the other users, but appears to be dictated by social norms.

Group areas were identified between adjacent people. They were used by neighboring users to perform the main activities like assembling tiles.

Location of storage areas could change dynamically and sat atop the personal and group areas. They were used to store currently unused puzzle tiles, taking the ownership properties of the area below them.

Proxemics divides human-human interaction spatially into zones.

In general, these results are supported by many studies done in the field of proxemics, introduced by Hall [1966]. It divides human-human interaction spatially into zones. Each zone has typical interactions associated with it (see Table 2.1). Note that these distances can vary from culture to culture. Latin cultures allow for closer distances, while Nordic cultures generally prefer larger distances.

Changes in territorial behavior can only be minor.

Knowledge of territoriality is important for analysis of any tabletop situation. This is especially true for PinguTouch, since it aims to maximize simultaneous interaction in the

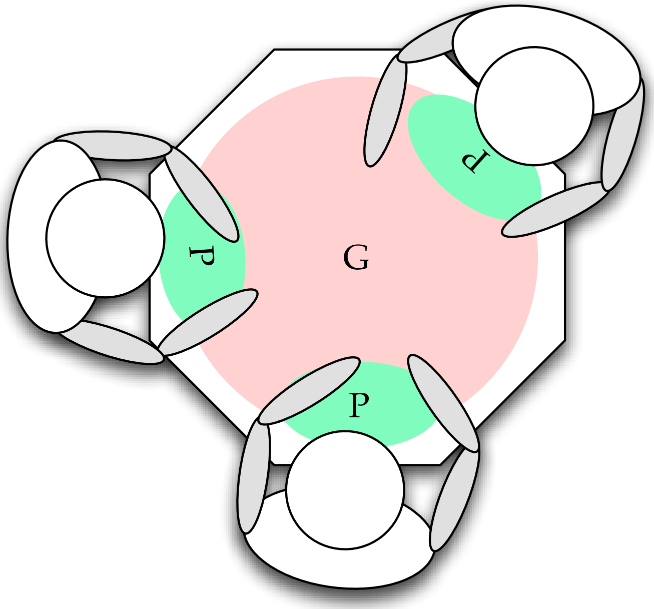


Figure 2.1: Tabletop territories. P indicates personal territories; G indicates group territory. Storage territories are not included.

group territory and to deemphasize individual actions in the personal territories. We are particularly interested if the game design can lead to players neglecting the social boundaries and reach into the personal areas of other players. However, since territoriality seems to be deeply ingrained in both human culture and nature, we can only expect minor changes in behavior for our game.

Zone	Distance	Typical behavior
Intimate	0.00m - 0.45m	Embracing, touching, or whispering
Personal	0.45m - 1.20m	Interaction among good friends
Social	1.20m - 3.00m	Interaction among acquaintances
Public	3.00m - ∞	Public speaking

Table 2.1: Proxemic distances and associated behavior. Adapted from Hall [1966].

2.1.2 Collaborative Coupling

Pairs should solve interfering routing problems.

[Tang et al., 2006] describes an application developed to observe collaborative coupling, i.e., the way people switch from performing tasks by themselves to working with others (see Figure 2.2). Pairs of users were given the assignment to solve a routing problem along a graph structure. To encourage negotiation, different sub-tasks designed to spatially interfere with each other were assigned to each of the two users.

The application included several tools to facilitate multi-user collaboration: users were able to lay lenses on top of the normal view, so that specific information is only visible within that area without interfering with the rest of the workspace. They could also use "shadow boxes" to duplicate the display of one area of the table to another.

Users fluidly change their cooperation style.

The participants were video recorded and the video feed was subsequently analyzed regarding the way people worked together. The changes between styles of collaboration were fluid and dynamic. The configuration of the users' focus was segmented into six separate coupling styles. At the same time, position arrangements were coded into seven positions. The coupling styles were correlated with the position arrangements, and, as would be expected, for tighter collaboration the physical closer or straight across arrangements were chosen.

The authors conclude that potentially many more different coupling styles could exist. Productive tabletop systems should support a variety of tools and concepts to support those styles and provide for fluid transitions between them. Furthermore, to reduce interference, the authors propose personal views that could be decoupled from the main display.

Research gives important baseline information.

The described research effort gives important baseline information about the behavior of cooperating users. Although the problem used for analysis was designed with spatial interference in mind, it also included many opportunities for users to act individually. PinguTouch aims to minimize such opportunities, thus focussing even more on

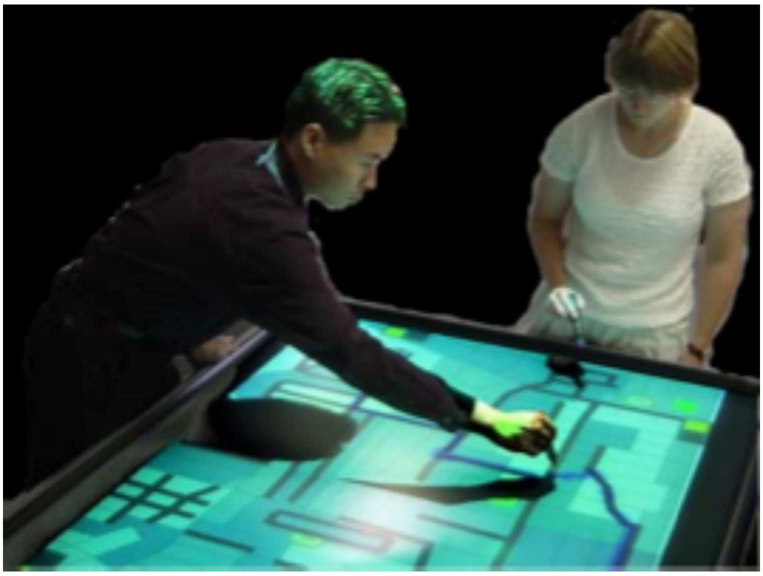


Figure 2.2: Investigating collaborative coupling; taken from [Tang et al., 2006]; reprinted with permission.

collaboration.

2.2 Multi-touch hardware

Since an inexpensive, scalable multi-touch screen is the major enabling component of this work, several approaches to multi-touch sensing will be discussed here.

2.2.1 DiamondTouch

The DiamondTouch table [Dietz and Leigh, 2001] allows for multi-user interaction and identification. Its technology is based on measuring capacitive coupling between the table and the user. Image generation is done by simple front projection onto the table surface.

DiamondTouch uses rows and columns of antennas carrying electric signatures.

Isolated antennas are embedded into the table surface in rows and columns. A transmitter drives a signal with a unique electric signature through each antenna. Should the hand of a user come near the surface, capacitive coupling transfers current from the nearby antennas to the user. These signals are then conducted through the user's chair to a receiver that can identify their origins by signal processing. Since each user has his own signal receiver, each contact area can easily be assigned to a specific user.

Disadvantages include difficult construction and requirement for conductivity.

A major disadvantage of this method lies in the production of the antennas and the custom circuitry needed to handle the electrical signals. Electrical engineering knowledge is needed, as there is no easy way to recreate this hardware using of-the-shelf components.

Some other limiting factors have to be kept in mind: users must always keep contact with their chairs, since only then can the signal be transferred to the receiver. If users touch each other, they become part of the same circuitry, and touch points of one user will be detected by both receivers. Furthermore, the system can only detect touch when a conducting material is used for contact.

2.2.2 Apple iPhone & iPod Touch

Apple hardware measures loss of capacitance along rows and columns of antennas.

The Apple iPhone and iPod Touch also measure capacitance. However, instead of including the user directly in the circuit, it detects touch by measuring loss of current to the user.[1]

The touchscreen consists of two layers of electrically isolated circuit paths, respectively called driving lines and sensing lines. The conductors in one path are arranged in rows, while the others are arranged in columns. Current is driven into the driving lines, one line at a time. At the intersection points, capacitive coupling transfers the current to the sensing lines. Should the user touch the screen, he draws off some of that current. This difference can be detected by the receiver attached to the sensing lines.

Most disadvantages of the DiamondTouch system also apply to this technique.

[1] See EU patent no. WO2005114369.

2.2.3 Holowall

The Holowall [Matsushita and Rekimoto, 1997] uses an optical approach for touch detection.

Holowall detects IR light reflection.

An infrared LED light source, an infrared-sensitive camera, and a projector are positioned behind the Holowall. The projector is used to back-project an image onto the wall. The LEDs emit infrared light towards the back side of the wall. If there is no object there, the light can escape. Should the user touch the wall or place an object against it, the light reflects back and is picked up by the camera. The system can then calculate position and shape of the touch. It can also detect barcode symbols to provide users with additional options regarding the device placed against the surface.

The same basic technology is used in the recently announced Microsoft Surface multi-touch table, except that multiple cameras are used to allow for a wider field of view at shorter distance to the surface.

2.2.4 Frustrated total internal reflection

Multi-touch detection based on frustrated total internal reflection [Han, 2005] works similarly to the Holowall system described in the previous section and is the technique employed by PinguTouch.

The hardware consists of three major parts: an acrylic sheet that is flooded with infra-red light by LEDs attached to its edges, a projector that back-projects onto it, and an infrared camera aligned towards it. A PC is used for processing the camera input and generating output.

FTIR multi-touch uses infrared LEDs in an acrylic sheet, an IR camera, and a projector.

In the unused state of the touch screen, the infrared light cannot leave the acrylic sheet through its top or bottom plane due to an effect called "total internal reflection". As long as the angle of the light and the interface between acryl and air is below a certain threshold, the vast majority reflects back inside of the acrylic sheet instead of leaving (see

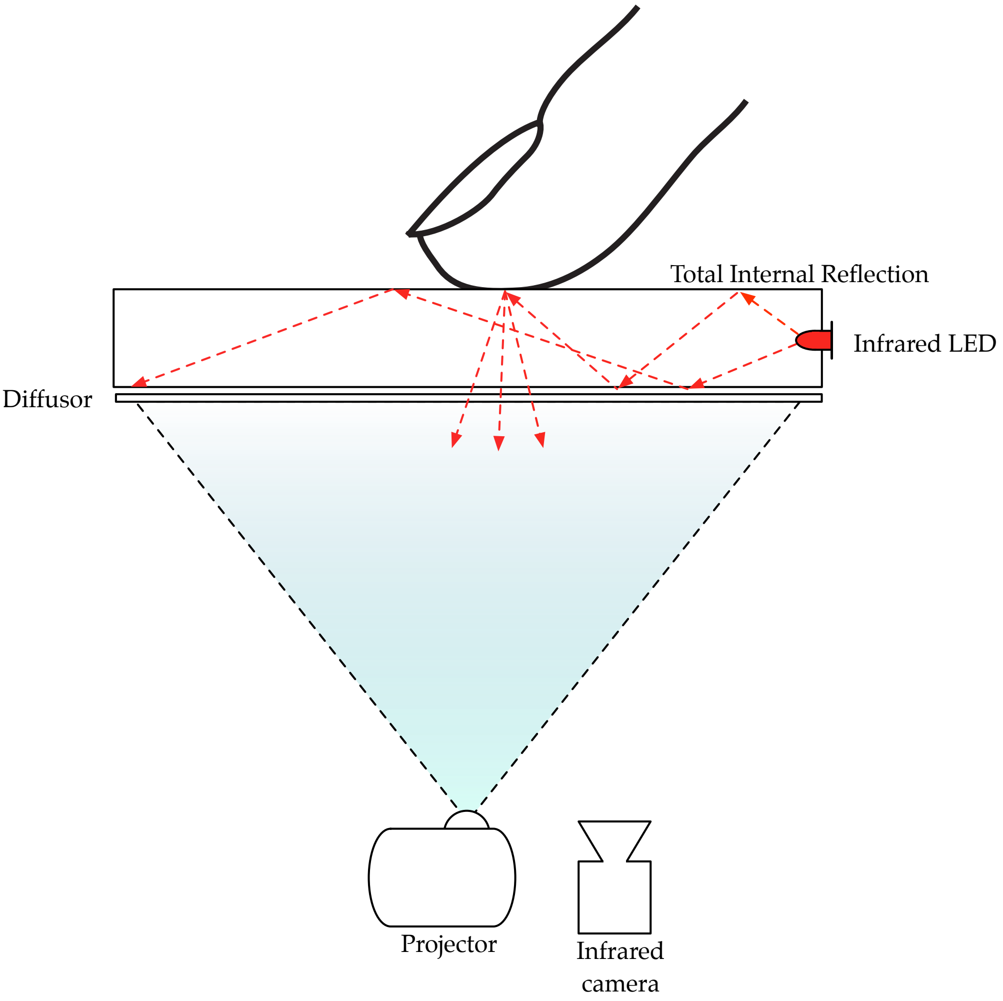


Figure 2.3: Concept of frustrated total internal reflection-based multi-touch technology.

Figure 2.3).

FTIR multi-touch detects spots of light leaving the table upon touch.

However, if another material—or, in this case, a user—touches the plane along which light is reflected, the reflection is frustrated and the light can escape the acrylic. This escaping infrared light is then detected by the camera as bright spots, or, "blobs" (see Figure 2.4). From those blobs, the coordinates of the touch point can be determined with the help of image processing software.

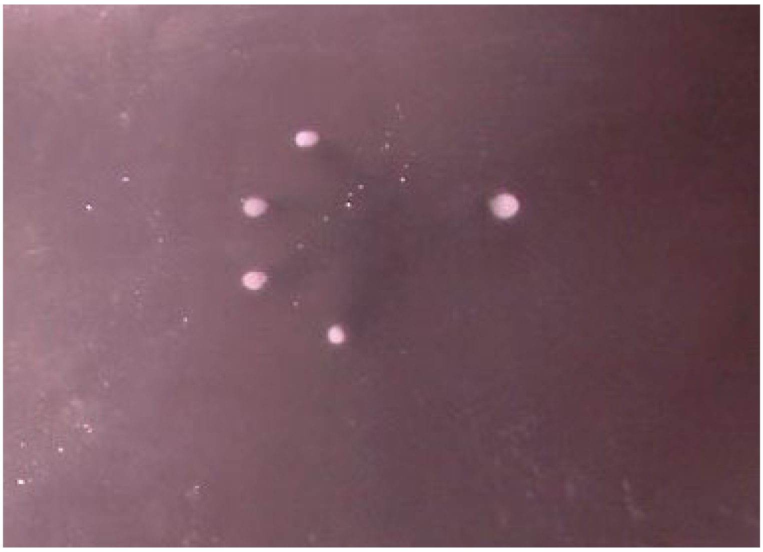


Figure 2.4: User touching the multi-touch surface as seen by an infrared-sensitive camera. All five touch points are clearly visible.

The resolution of the touch input is limited only by the resolution of the camera and the necessary computing power for coordinate extraction. In the age of high-performance multi-core CPUs and 5 megapixel cameras, this solution is elegantly scalable to even whole walls.

2.3 Tabletop applications involving complex hand gestures

The following applications were designed to explore new input gestures enabled by multi-touch technology. Since one goal of PinguTouch is to offer complex input gestures, we investigated these previous ideas for possible adaptation.

2.3.1 CollabDraw

A multiuser drawing and photo organization application was studied in [Morris et al., 2006]. Users were able to rotate and move photos and draw simple strokes onto a DiamondTouch table. To erase ink, users could perform a wipe-gesture with the palm of their hand.

Cooperative gestures consist of gestures of multiple users interpreted as a single command.

The stand-out approach in this work is the concept of cooperative gestures: “Cooperative gestures are interactions where the system interprets the gestures of more than one user as contributing to a single, combined command.” This is used as part of the “modify ink” gesture: pairs of users have to partner by holding hands and touching the tabletop. Then, one partner can draw on the surface by sliding a single digit along it. The other partner can place two fingers on the surface, and according to their distance, line thickness changes. Also, line brightness can be influenced depending on how hard the user presses.

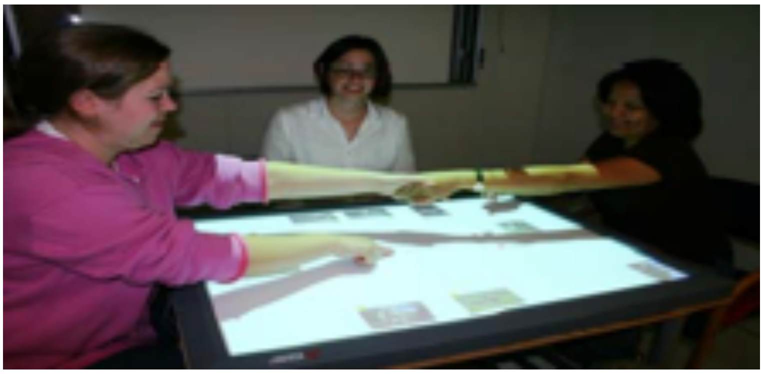


Figure 2.5: Two users performing a partnering gesture in CollabDraw; taken from [Morris et al., 2006]; reprinted with permission.

Problems: unintentional global commands, awkward social situations.

Some of the gestures that are also usable by a single user are overloaded when performed by all users at the same time: The “erase ink”-gesture of wiping over the table works as a “clear canvas”-command when performed as cooperative gesture. The “arrange photos”-gesture, that neatly lines up the photos, can work similarly on a global level. This led to frustration and confusion of users when the global commands were invoked unintentionally. Additionally, some of the cooperative gestures were described as akward to execute: for the completion of several actions, the users had to be in tactile contact with each other (e.g., by holding hands), which led to awkwardness in their current social situations.

Cooperative gestures also promote close spatial cooperation, but often awkward to execute.

Since the hardware we used is not able to identify users, PinguTouch is not designed to require direct contact between the players to perform actions. However, our idea to promote close spatial cooperation is similar to that of cooperative gestures. We expect that the desire for close cooperation emerges naturally from our game design, thus avoiding the discomfort described by the players.

Gestures involving sustained movement of the hand, similar to the wiping gesture, are also hard to integrate into our design due to our decision to use quasi-modal gestures. The repetitiveness of the motion would quickly lead to imprecision and user fatigue.

2.3.2 RoomPlanner

RoomPlanner can be used to plan room furniture layout.

In [Wu and Balakrishnan, 2003], a collaborative co-located workspace was examined by using the RoomPlanner application to plan room furniture layout on a DiamondTouch table. This scenario presented interesting opportunities for research, e.g., efficient menu navigation, privacy concerns, and the concept of object ownership.

Supports several whole-hand gestures.

The system was designed for two users, each working from one of the long edges of the table. One supported gesture involves placing the edge of the hand vertically on the table to sweep aside several furniture pieces at once. Another gesture allows a user to obtain general information about objects by placing the hand horizontally above it. Furthermore, he can then tilt the hand slightly away from him, enabling the system to project any private data onto the palm of his hand.

Supports two-hand gestures.

The application also supports gestures involving two hands: when multiple furniture objects are enclosed between two vertical hands, movement of the hands towards or away from each other could compress or expand the empty space between the furniture. Users can also copy sections of the room into private editing planes by enclosing them with corner-shaped hands.

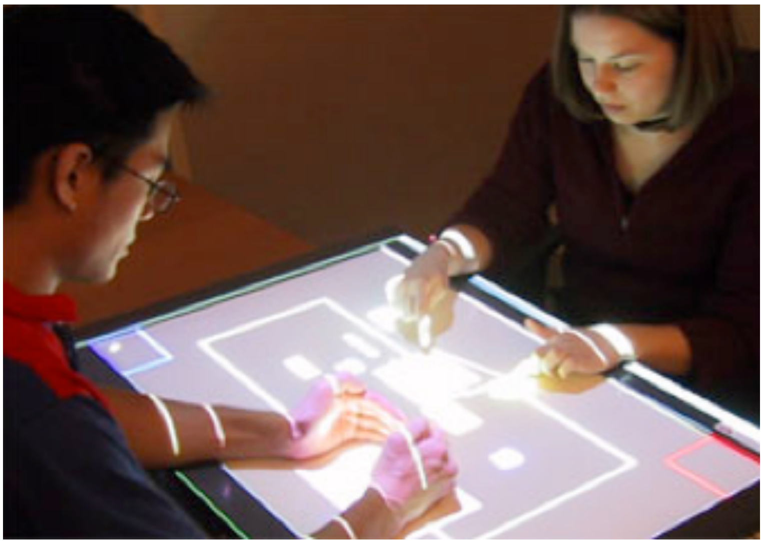


Figure 2.6: The RoomPlanner application; taken from [Wu and Balakrishnan, 2003]; reprinted with permission.

Gestures difficult to adapt to our concept and technology.

Although RoomPlanner implements several complex gestures, adaptation for PinguTouch is impractical. Differentiating between horizontal and vertical placement of the edge of the hand would be impossible, as players are encouraged to move freely around the PinguTouch table. Similarly, two different users placing their hands orthogonally to each other could easily be detected as a corner-shaped hand in our highly cooperative scenario.

Detecting gestures involving a tilted hand would also be unreliable with our FTIR hardware, since we cannot depend on electric sensing to detect proximity, as is possible with the DiamondTouch table.

2.4 Cooperative interactive tabletop games

2.4.1 Modified commercial games

Adapted commercial games were reported as more engaging on a tabletop.

Tse et al. describe attempts to enable commercial desktop games for tabletop interaction [2006]. For behavioral investigation, the authors modified the games "Warcraft III" and "The Sims" to support two-player co-located gaming, utilizing both multi-touch and audio input (see Figure 2.7). The applications were reported to be more engaging and entertaining than their original desktop versions, but no more detailed study was presented.

Modified Warcraft 3 most similar to PinguTouch.

The modified Warcraft 3 game is closest to the system described in this thesis. It has a pronounced real-time component and users have equal control over many areas of the table. However, since the original Warcraft 3 was not developed with a co-located gaming scenario in mind, players of the tabletop version could not act truly independent of each other. They had to take care not to issue a command while the respectively other player was also working, or else the system would confuse the input. This issue practically restricts the game to two players.

No close spatial cooperation.

This modality of the interface helps to enforce collaboration, but close spatial cooperation is not encouraged by the game design.

2.4.2 SIDES

SIDES players have to find a common strategy to lead a frog to the goal.

SIDES [Piper et al., 2006] is a game developed to further social skills in people with Asperger's Syndrome, an autism spectrum disorder. The goal of the game is to lead a frog across a tabletop setup using placable virtual lily pads that control the direction the frog takes next. Players have to devise a strategy that includes the tiles available to each player, and take turns in placing them (see Figure 2.8). The hardware is based on a DiamondTouch screen that supports multi-touch functionality including user identifica-

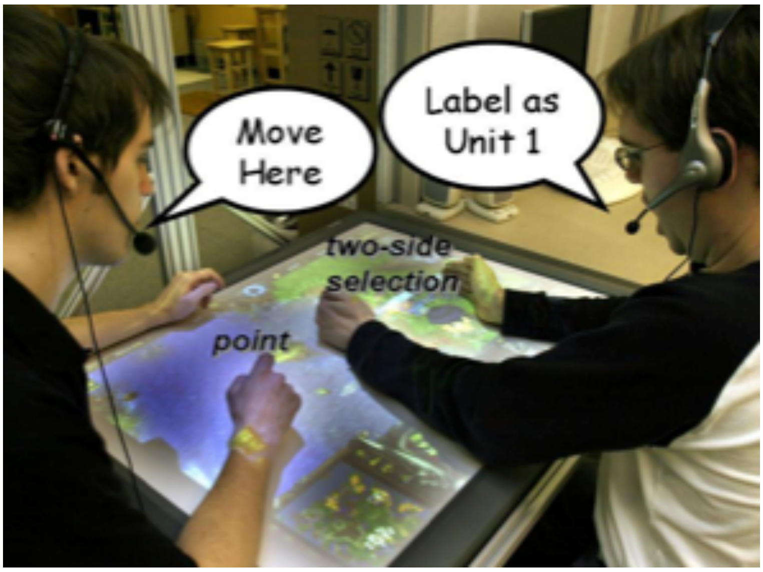


Figure 2.7: Modified Warcraft III game; taken from [Tse et al., 2006]; reprinted with permission.

tion as described in Section 2.2.1—"DiamondTouch". This allows enforcement of players adhering to the correct order to teach group behavior.

Players have to take turns.

Although SIDES is cooperative in nature, it is decidingly different from PinguTouch: due to its turn-taking nature, the final version of SIDES barely uses the multi-touch aspect of the system. It was an explicit design-goal to always give the players time to discuss strategy, so there are no time-critical situations.

2.4.3 Jam-O-Whirl

Jam-O-Whirl uses drums and turntables as input.

The Jam-O-Whirl system [Blaine and Forlines, 2002] consists of a set of games focussed on collaborative musical composition. A tabletop setup is augmented with four stations consisting of a drum, a turntable, and a directional audio speaker each. Visual feedback is projected onto the tabletop surface and the environment surrounding it.

One game called "CircleMaze" requires players to align

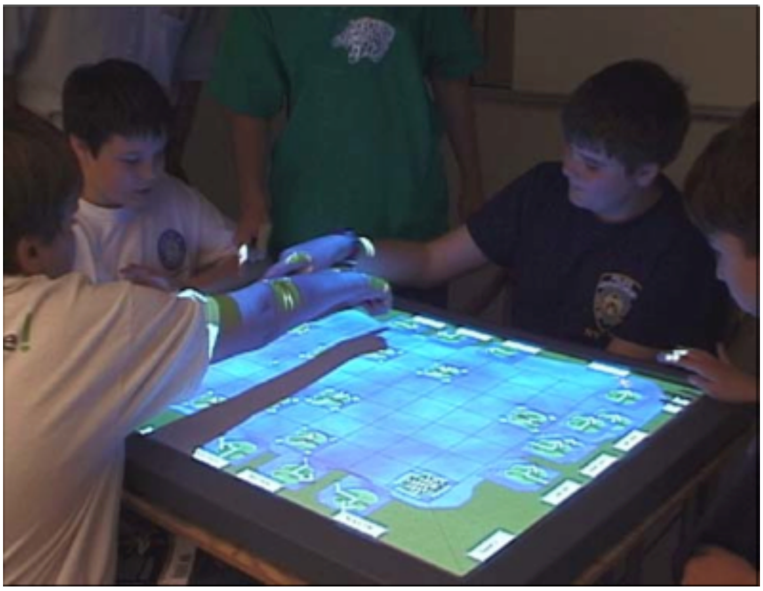


Figure 2.8: Children interacting with SIDES; taken from [Piper et al., 2006]; reprinted with permission.

concentric rings by rotating their turntable. The rings contain pathways, that, when aligned correctly, connect to each other. A ball projected onto the surface can follow these pathways, and has to be guided from the perimeter of the table towards its center. This requires coordination between all players.

The game requires coordination between all players.

"Hip Hop", another game developed for the Jam-O-Whirl table, requires players to match motives on cards displayed on the table by using their turntable station. Again, all players must cooperate to succeed.

While this system was designed from ground-up with collaboration in mind, the interaction with the game world by using the drums and turntables is less direct than the touch-based system we propose, since input and output are spatially decoupled. Although the instruments stay true to the musical theme of the game, they also restrict the users: the turntables are fixed at their position, so every player has his own set of input devices, completely avoiding close spatial cooperation.

The fixed input devices prevent close spatial cooperation.

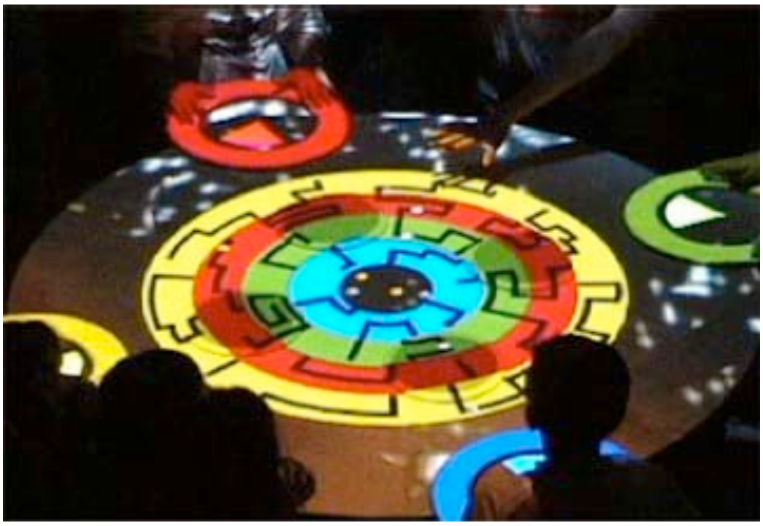


Figure 2.9: The Jam-O-Whirl system running CircleMaze; taken from [Blaine and Forlines, 2002]; reprinted with permission.

2.4.4 Magic Land

The Magic Land system by Qui et al. [2005] uses augmented reality to allow the integration of real-time video captured humans and virtual characters in a tabletop scenario.

Players can integrate a recording of them with other avatars in a virtual world.

Players can record a virtual avatar of themselves in a specially equipped recording room. The data is represented in form of a cube that the user can carry with him. He can use this cube to load his avatar onto a table by placing it into a slot. In a similar way, he can select a virtual landscape by putting one of several available puzzle pieces into another slot.

With a head mounted display, he is then able to see a virtual scene integrating computer controlled characters, his own avatar, and a real-time recording of another person in the recording room. The user can direct his avatar by moving the cube along the table. Should it come close to another character, the two will interact in a predefined way.

The system is more art piece than game.

One could imagine multiple persons using this system to cooperatively create a virtual stage play. However, there

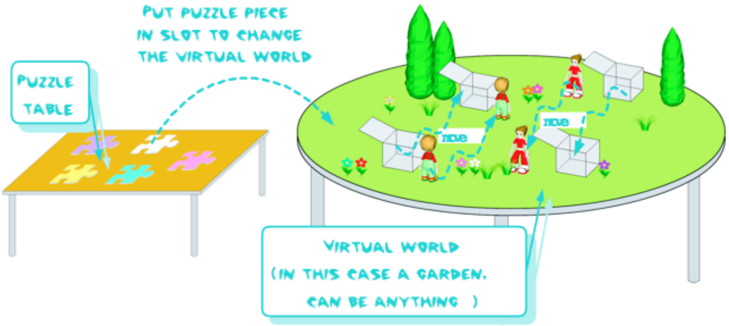


Figure 2.10: Illustration of the Magic Land system; taken from [Qui et al., 2005]; reprinted with permission.

are no concrete rules or goals involved and interaction with the game world is limited to moving the characters. We believe that in the current iteration, the system is more art piece than game.

2.5 Discussion

The systems reviewed here can be classified along the three criteria by means of which PinguTouch was developed:

Real-timeness, perceptibility of co-location, and sophistication of input gestures form a design space.

- **How noticeable is the real-time aspect of the system?** Does the system employ a concept of time passing, i.e., can there be system-initiated events that the user has to react to within a specific timeframe? Low indicates that the system does not create any events. High indicates that many events happen, following a complex rule set. To better differentiate between systems, any gradation in between is possible.

- **How noticeable is the the co-location of multiple users?** How much would the experience change if the application would be transformed to a networked setup? Low indicates that the application does not specifically encourage behavior that exploits the co-located nature. Medium indicates that the application encourages discussion among the users. High indicates that the application encourages the users to act physically closely together.

- **How sophisticated are the supported input gestures?** Low indicates the application only supports simple pointing. Medium indicates that the application supports one or few more complex gestures. High indicates that the application supports several more complex gestures.

The design space described by these criteria is visualized in Figure 2.11. Although the task-based applications we reviewed in this chapter are included for sake of completeness, ordering them by a different design space might be more revealing.

PinguTouch occupies own spot in design space.

While all of the cooperative tabletop systems discussed here are viable applications in their own right, none follows the same design goals as PinguTouch.

PinguTouch achieves only medium gesture complexity.

Even though our initial game proposal called for highly sophisticated gesture support, we had difficulties integrating many complex gestures into the game due to the employed technology and contradictions to our other design goals (cf. Section 2.3—"Tabletop applications involving complex hand gestures"). Additional research has to be performed to find other ways to increase gesture complexity.

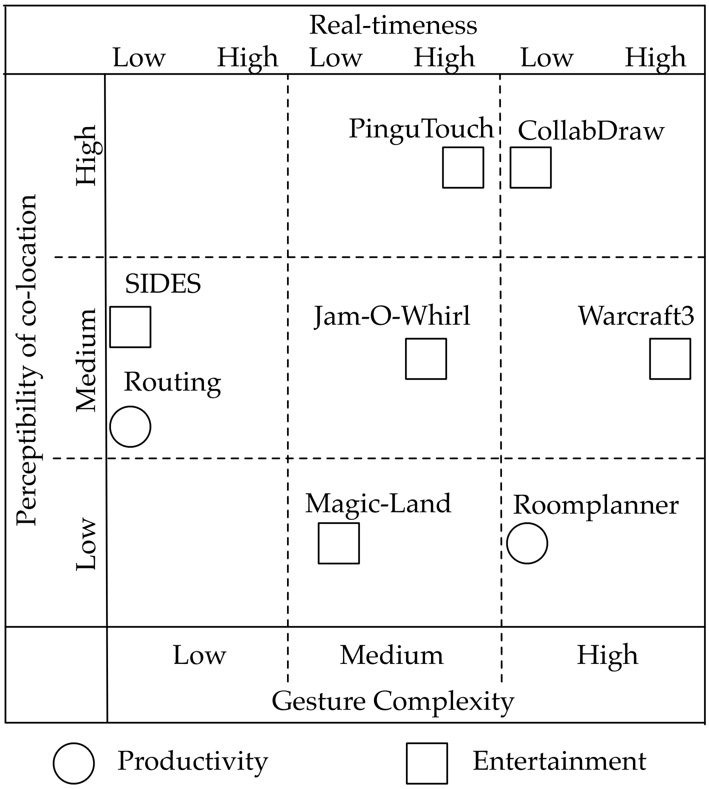


Figure 2.11: A design space for cooperative tabletop applications.

Chapter 3

Designing the game

> *"Everyone who has a computer fancies himself a game designer, just as everyone with a guitar wants to be a rock star. There is nothing wrong with that if you remember that success is a long, hard road."*
>
> *—David Crane, designer of Pitfall,*
> *in K-Power Magazine, April 1984*

Apart from the more basic requirements for cooperation as discussed in Section 1.1—"Cooperative gaming", we propose the following three criteria for a game to be suitable for a tabletop multi-user experience:

The game must offer equal accessibility for all players, multiple points of interest, and possibilities for high-semantic gestural interaction.

- **There must be multiple points of interest at the same time.** A game with only a single point of manipulation at any one time is obviously not suitable for adaptation to a cooperative multi-touch scenario. Any game with only one central controllable character would be excluded by this requirement.
- **The game must offer equal accessibility to all players.** Classical tabletop games are designed in a way that allows all players equal access to the game board. In many cases, this is accomplished by fashioning the fields on the board around a circular way along the outer edges of the board. The board is usually rectangular, allowing several players to sit around it.

Since the game developed as part of this thesis should afford a similar way of interacting, the same design guidelines apply. The players should have a common view. Navigation through a game world in a way that involves multiple players is hard to synchronize and best be avoided. One player might want to change the view, while another is still performing an action at a spot that might be unreachable after the view change. This requirement excludes any game that is based on a first-person perspective of the player's character.

- **There must be the right level of semantic involved in the interactive possibilities.** Since actions must be mapped to two-dimensional hand gestures that make sense to the user, those gestures should be clearly distinguishable and relevant to its semantics, yet not be too complicated. Although this criteria should not be hard to fulfill, certain types of games are more amenable to recognizable gestures than others. Games that rely mostly on navigational tasks, e.g., racing games, are not particularly suitable.

Our users are young, inexperienced, short-time gamers.

In addition to these guidelines, the target demographics also played an important role in our design considerations. Since the game was going to be deployed at the Industrion museum, we were able to get some user information from the administration. It was estimated that about 40% of their visitors are below the age of 18, with most being between 8 and 13. Few of the them were expected to be experienced gamers, so the game mechanics had to be simple. Understanding the game should not necessitate knowledge that is common among frequent gamers, but might be lacking among the general population. Furthermore, since the game is part of a larger exhibition, users should not be required to play for more than about 10 minutes at most to be able to experience the whole game.

3.1 Finding a game concept

Several game concepts based on existing titles were considered as a basis for our work. One of the first ideas was a reinvention of the classic Pong game(see Figure 3.1a). It was envisioned that users could use their hands to create paddles, thus being able to easily rotate and move them along the table. Multiple balls could be in the game, depending on the number of hands on the table. Teams could naturally emerge, grouping players from each side of the table together.

While the idea would be easy enough to implement, it was somewhat lacking in novelty. Only one type of gesture would be involved, and the ability to spontaneously add paddles anywhere on the table posed a hard design challenge to balance such a game. For similar reasons, the idea of a table football game, where the fingers of the players could represent football players, was rejected.

A variation of 'Pong' was deemed too unoriginal.

Another idea was based on the game "The Incredible Machine", produced by Dynamix[1] in 1993 (see Figure 3.1c). Players could build contraptions out of a large set of predefined parts to fulfill senseless goals, e.g., moving a ball into a basket or firing of fireworks. The concept of building some sort of machine in collaboration is intriguing, but including meaningful gestures for building them are hard to conceive. Furthermore, the implementation would have to be built nearly from scratch, since only the beginnings of an open-source version are available.[2]

'The Incredible Machine' had no source code available.

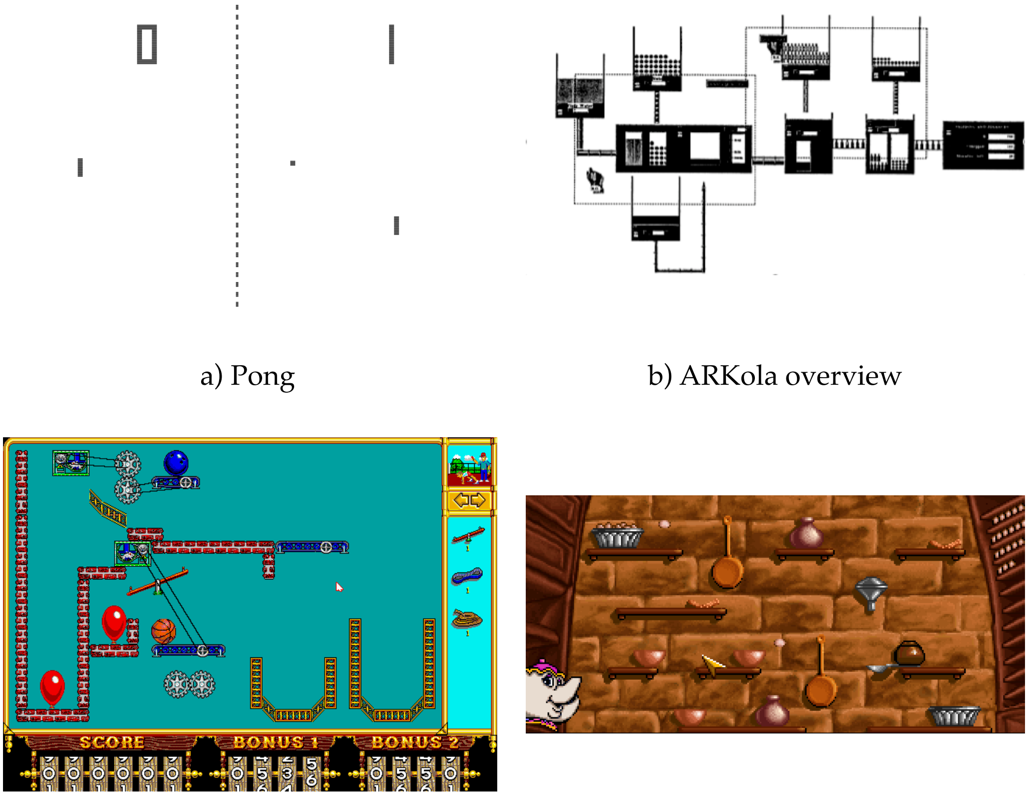
From the idea of building a machine, the leap was made to the player actually performing the processing tasks themselves. The fundamental idea consisted of a pipeline of material that must be processed in multiple steps within a specific time limit for optimal efficiency. As an example, the ARKola bottling plant simulation as described by Gaver et al. [1991] was used (see Figure 3.1b).

Similarly, a game based on the movie "Disney's Beauty and the Beast" developed by Infogrames Entertainment[3] in 1994 contained a puzzle that required the player to guide

[1] The company was disbanded in 2001. No information on any developed products could be found on any officially associated webpage.

[2] See http://tpm.seul.org/.

[3] Several subsidiaries of the company have been renamed under the Atari brand. No product released before 2002 can be found on any webpage officially associated with the company.



a) Pong

b) ARKola overview

c) The Incredible Machine

d) Beauty and Beast

Figure 3.1: Images of games used as inspiration. **b)** taken from [Gaver et al., 1991].

jumping eggs through a course of kitchen instruments into a baking form (see Figure 3.1d).

Using 'Lemmings' was based on the idea of processing a material pipeline.

This type of time-critical resource management evolved into the idea to create a game similar to "Lemmings". Lemmings is a puzzle-game produced by DMA Design Limited[4], that was popular in the early 90's.

Lemmings requires the player to lead helpless creatures through a 2D landscape.

The game is divided into levels. Each level presents a side view of a two-dimensional landscape and contains at least one entry point and at least one exit. Little creatures, the

[4] The company was renamed to Rockstar North in 2002. No information on any projects developed before 2001 could be found on their official webpage.

lemmings, drop out of the entry points and begin to walk in a fixed direction along the landscape. When they hit an obstacle, they turn around and walk along the other direction. Thus, by themselves, the lemmings are quite incapable of navigating through the hardships of life (see Figure 3.2). This is where the player comes in.

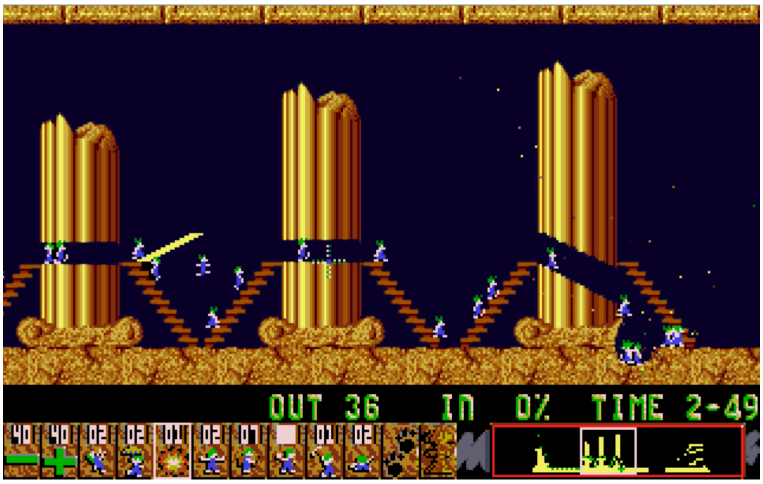


Figure 3.2: A screenshot of the original Lemmings game.

The player can assign abilities to lemmings.

The goal of the game is to lead the lemmings towards the level exit without taking too many casualties in the process. Typical ways for lemmings to die include falling out of the level, falling down too far, and walking into a trap. The player can assign abilities to the lemmings to overcome the obstacles presented in the level. Some typical abilities include digging, blocking of other lemmings, bridge building, and climbing. To assign an ability to a lemming, the player must first enter the corresponding mode by clicking on the ability icon. He must then click on any lemming he wants to give that ability to. This is a typical verb-noun interface similar to those from task-based programs, e.g., drawing applications.

At first glance, only the first of the three above mentioned criteria applies favorably to this game: all of the lemmings currently in the game (usually more than a dozen) allow for interaction, so that there are always enough tasks to occupy multiple players. Additionally, other elements in the game world apart from the lemmings could be made interactive.

Equal accessibility, however, is hindered, since the game would only be viewable correctly from one edge of the table. Also, while the general game idea to assign abilities to lemmings to overcome obstacles is open enough to provide enough space for different gestures, the verb-noun interface is very clumsy. It does not use any of the possibilities that multi-touch technology provides to communicate more semantics than simple coordinates. Fortunately, these two aspects could easily be conceptually resolved.

A tabletop layout can be achieved by transforming the game world to a circular view.

A tabletop layout is achieved by transforming the game world from the linear to a circular view. In that sense, the previous "down"-direction leads towards the perimeter of the circle, and the "up"-direction towards the center. Thus, players are always able to sensibly interact with the area in front of them, independent of their position around the table.

Gestures should be applied to the environment, not to the lemmings.

Since devising meaningful gestures representing the actions assignable to the lemmings proved difficult, control of the individual lemmings was replaced by a gesture recognition system that does not give abilities to the creatures themselves, but changes the environment directly. Those changes to the environment are performed in a quasi-mode fashion, so that the cooperation of multiple users is required to most efficiently lead the lemmings to the goal (cf. Section 1.1—"Cooperative gaming").

3.2 Introducing: PinguTouch

As a starting point for our work, the game "Pingus" was chosen.[5] Pingus is a recreation of Lemmings, featuring a completely self-designed code base and updated artwork. Its GPL (GNU General Public License)[6] enables anyone to access and modify the source code, and reuse any data distributed with the game. The basic game mechanics are the

[5] Available at http://pingus.seul.org.

[6] See http://www.gnu.org/licenses/old-licenses/gpl-2.0.html.

same as in Lemmings, except that the protagonists are little penguins aptly called “pingus”. Hence, we decided to name our game “PinguTouch”.

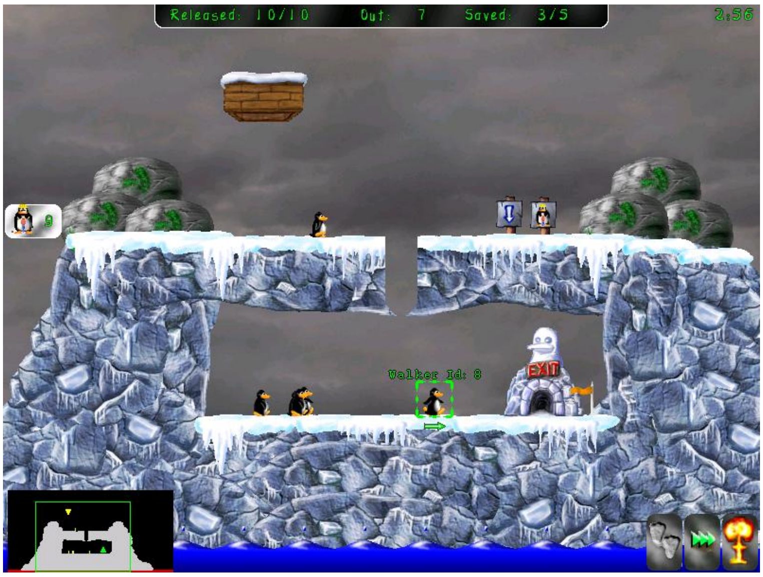


Figure 3.3: A screenshot of the original Pingus game; taken from `http://pingus.seul.org`.

3.3 Appearance design

For total equality amongst players, a circular form of the game display would be optimal. However, this would lead to strong disadvantages in other areas.

Instead of a circle, a regular octagon was used for output.

From a usability point of view, gestures performed in a circle are less intuitive: a gesture that looks straight from the users point of view, e.g., placing the edge of the hand on the table, would actually be curved in the game world.

On the technical side, implementing true circular output is hard starting with the available code from the original Pingus. Since there is no practical pixel to pixel mapping from a rectangle to a circle that keeps the proportions intact, most of the graphics subsystem, collision detection, level editor, and probably many other areas that weren't even considered would have to be re-implemented.

To avoid these concerns, it was decided that instead of a circle, a regular octagon would be used for output. This modification requires less work to implement compared to the circular form. For the application of gestures, a perspective change only happens when the user's gesture crosses from one segment of the octagon into the next. Observations revealed no instances of users being confused by this behavior.

3.4 Interaction design

Flicking, creation of bridges, and pulling of levers are possible gestures.

Three different gestures were devised for the initial design: the flicking of pingus, the creation of bridges or blocks, and the pulling of levers. The gestures were developed to be easily distinguishable from each other by users as well as by the system itself.

3.4.1 Flicking of pingus

Users can perform a flicking gesture on the pingus.

With the flicking gesture, the player is able to prod a pingu towards any direction. It is performed by flicking a finger over the pingu in the target direction (see Figure 3.4).

3.4.2 Opening doors

By keeping a lever pulled, users could open doors.

This gesture allows the pulling of chained levers with a finger to open doors in the level (see Figure 3.5). To further cooperative behavior, the player must keep the lever pulled down for the door to stay open.

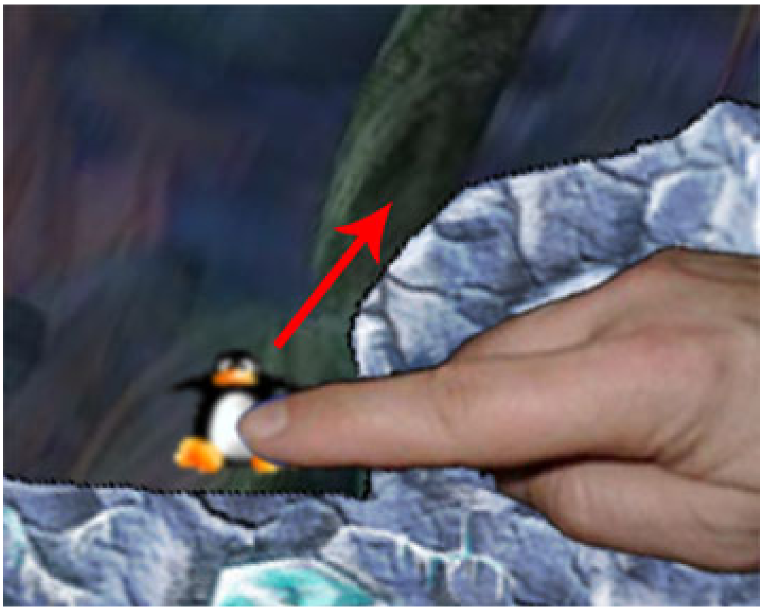


Figure 3.4: The flicking gesture in action.

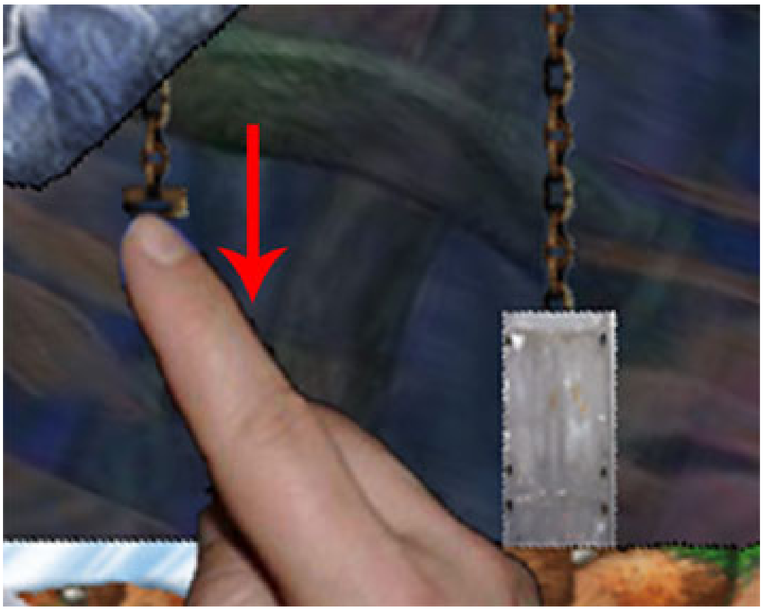


Figure 3.5: The opening gesture in action.

3.4.3 Blocking and bridging

Placing the edge of the hand on the table creates a block.

By placing the edge of his hand on the table, the player can create a block at that place within the level. This block can either be used as an obstacle to obstruct the path of the pingus or as a bridge-way to reach previously unreachable areas (see Figure 3.6). As soon as the player removes the hand, the block disappears.

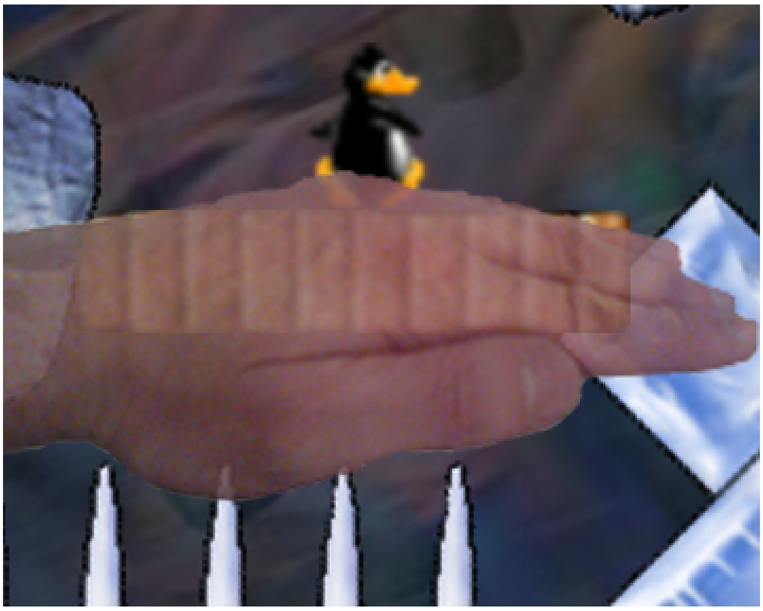


Figure 3.6: The bridging gesture in action (montage for better clarity).

3.5 Level design

Difficulty, opportunity for cooperation, and visual appeal have to be balanced.

PinguTouch consists of only one level. The output octagon has a radius of 512 pixel at its peaks, so level space is very restricted due to the non-changeable view. Any level design has to balance difficulty with necessity of cooperation between players, while still keeping some visual appeal.

The level used by the first deployed version of PinguTouch is shown in Figure 3.7.

The area marked as (1) is the starting area. It is enclosed on both sides, so that players are not under stress from the start but have time to become comfortable with the input possibilities. They can either flick the pingus or make a bridge to transport the pingus across the ledge.

(2) denotes an area where players are forced to act if they do not want to lose pingus. They must block off the left side of the area, while transporting the pingu up the ledge to the right. From there, they must pay attention that the pingus cross the chasm at (3). The gap is designed to be rather hard to cross, so that one person will find it difficult to control the pingus in (2) and (3) at the same time.

(4) introduces levers. Once the players open door (A) with the correct lever, the pingus walk towards area (5). This area is designed so that it is more comfortable to coordinate with another player to open doors (B) and (C) by pulling the respective levers, rather than one player doing it all by himself.

Finally, (6) marks the goal area. One player must take care of moving the pingu into the goal, else they will simply walk back into the other direction and fall down the gap.

3.6 Designing an exhibit

Pattern language describes exhibit design experience.

Besides the low-level game design, special considerations had to be made to make the game suitable as an exhibit. In this regard, we could build on previous experiences documented by Borchers [2001] in form of a design pattern language for interactive exhibits (see Figure 3.8). Design patterns are generalized descriptions of problems and well proven solutions, originating from the field of architecture [Alexander et al., 1977]. Large-scale, abstract patterns can reference small-scale patterns, thus creating a language in form of a hierarchy.

Many design patterns easy to integrate.

Many of the applicable patterns, e.g., COOPERATIVE EXPERIENCE, INVISIBLE HARDWARE, INNOVATIVE APPEARANCE, and SIMPLE IMPRESSION, emerged directly from

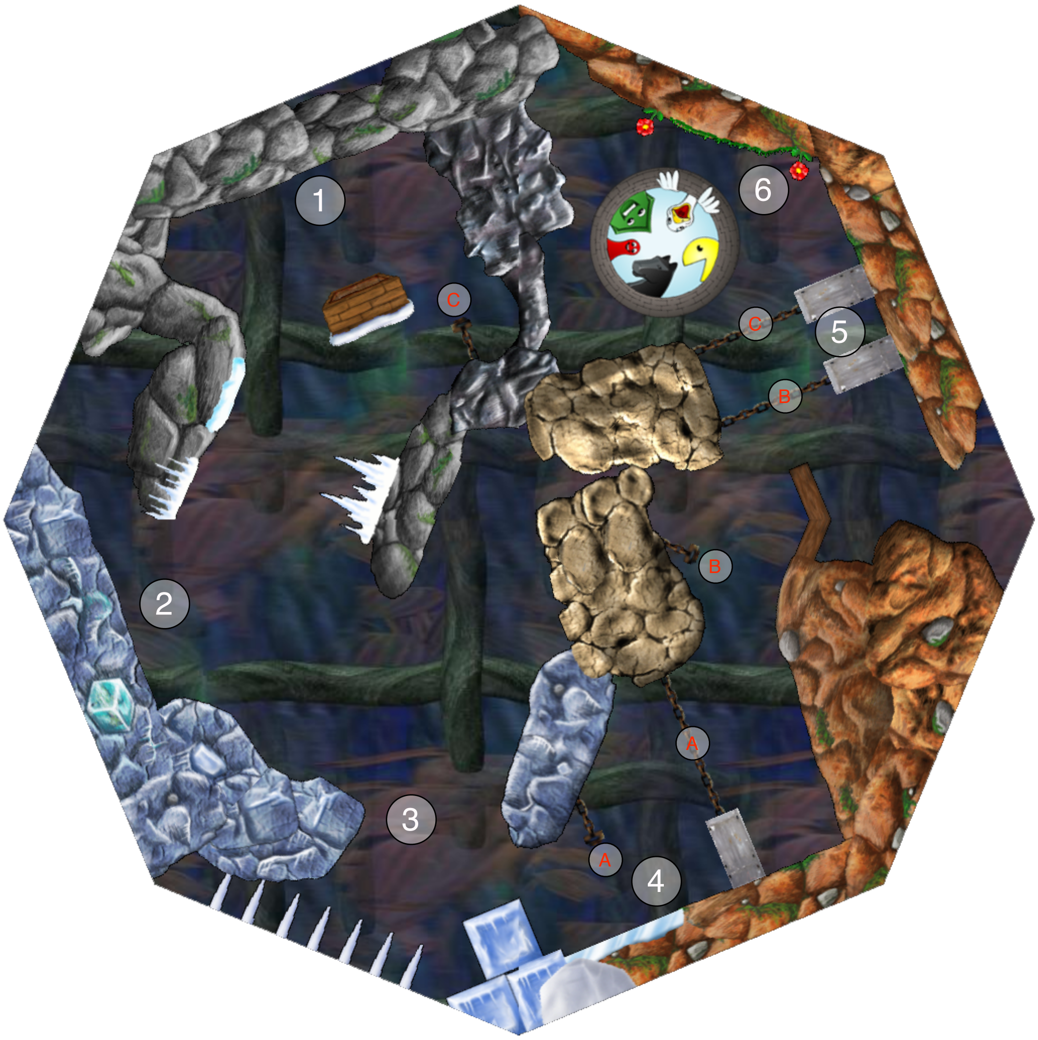


Figure 3.7: Initial level layout with description of areas.

the concept of a cooperative tabletop multi-touch game. Several others, e.g., LANGUAGE INDEPENDENCE, CLOSED LOOP, ATTRACTION SPACE, and EASY HANDOVER, could easily be integrated into the system design. Since the design consideration of exhibits are similar to that of casual games, we feel confirmed in our hypothesis that multi-touch technology lends itself well for the casual gaming market.

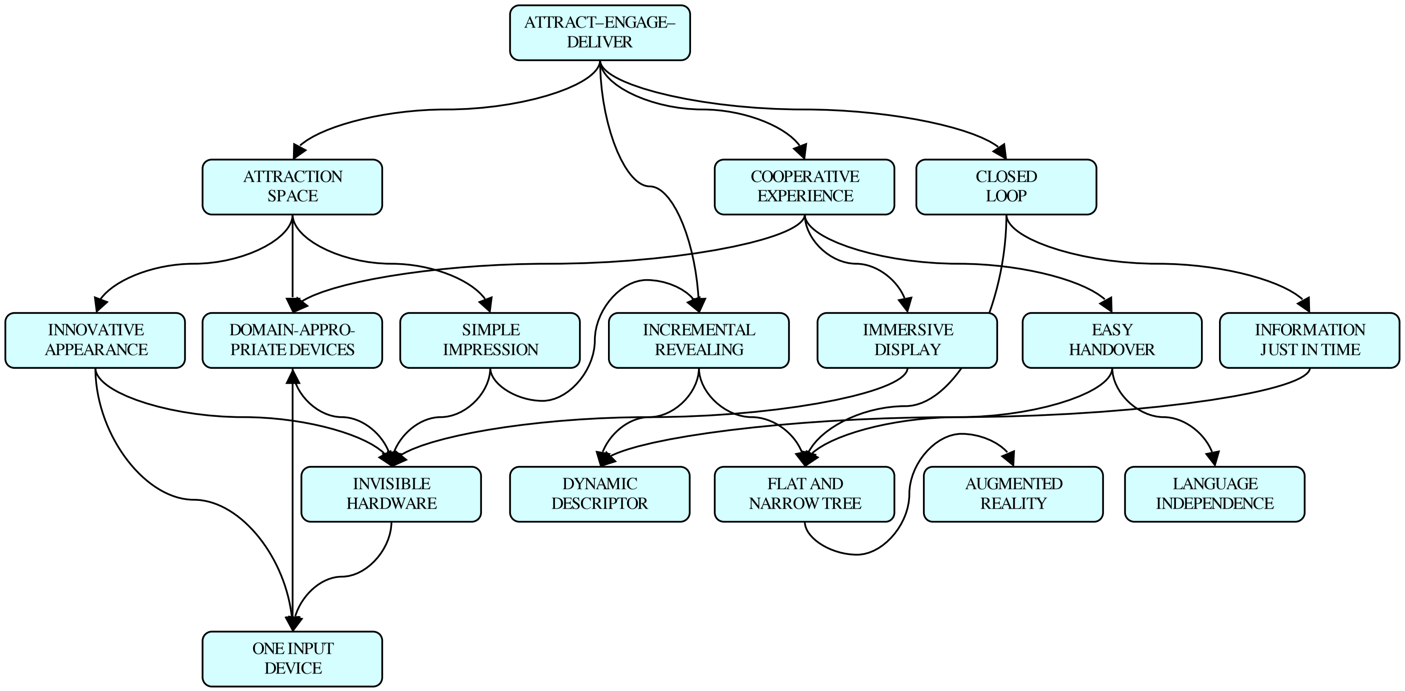


Figure 3.8: The design pattern language for interactive exhibits; taken from [Borchers, 2001]; reprinted with permission.

Chapter 4

Refining the game

"Just ship it! We'll fix it with a patch..."

—Overheard during many game publisher meetings

After deployment of PinguTouch at the Industrion museum, we had the opportunity for detailed user studies in the field. During user observations and interviews, several issues in the game design that hindered enjoyment were identified and resolved.

4.1 User studies

Ten groups of players were observed, of which three were interviewed.

We observed about ten groups of users playing the game during an afternoon at the museum. Additionally, members of three groups allowed us to interview them. The first interview was with a family of four, the second interview was held with two children of about age ten, and the third interview was conducted with a woman in her forties. Written permission to make recordings of the interviews was granted by the interviewees or, in case of children, their parents.

The interview was exploratory in nature, so the questions were very open ended. They were intended to obtain opinions about the cooperative nature of the game, reveal problems, as well as gather hints towards possible avenues to explore in further evaluations (cf. Section 6—“Evaluation”). Typical questions included ”Can you tell me how you felt about playing the game?”, ”Can you tell me how you felt when playing the game with others?”, and ”Can you tell what you liked or disliked about the game?”.

Users greatly liked cooperation, tabletop setting, close interaction.

The general consensus was that users greatly liked the concept of a cooperative game and indicated it was a novel idea. The tabletop setting was felt as a good direction for further development of interactive gaming. The first two groups explicitly stated they like cooperative gaming better than traditional competitive gaming. The woman interviewed in the third group found closely interacting with others, even strangers, the best part of the game.

4.2 Bridging gesture

To compensate for imprecise touch detection, collision detection of blocks was made more lenient.

We observed multiple times that users perceived problems with the bridging gesture. The gesture detection is unable to detect the exact position and angle of the whole hand, thus, the created bridge is rarely at the exact spot with the exact angle as intended by the user. To make matters worse, the user cannot see the actual location of the bridge, since it is being blocked by his own hand.

Since gesture detection cannot be notably improved with the technology employed, collision detection regarding bridges was made more lenient instead. Changes were made so that the pingus could walk steeper on bridges than they can on regular ground, and can jump a few pixels if the beginning of the bridge is not exactly lined up with the ground.

When the gesture was performed in an area where pingus were already active, chances were high that the bridge got created right on top of one or several pingus. This led to the pingus being stuck in place, unable to move at all. This behavior was changed, so that the stuck pingus would jump on top of the bridge.

For better visual feedback, a translucent glow was added around the blocking graphics to indicate the extents of the bridge without actually appearing as solid material (see Figure 4.1).

Blocking graphics were enhanced for better visibility.

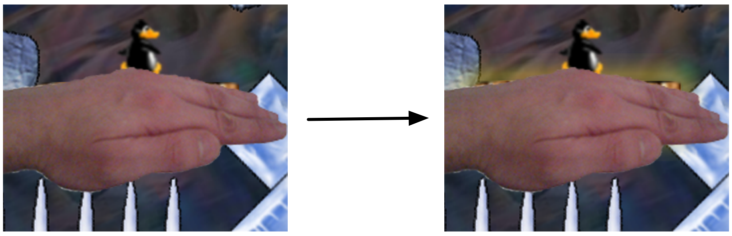


Figure 4.1: Left: Original bridge appearance. **Right:** Revised bridge appearance (montage).

4.3 Switches

Users were often unable to recognize the switches of doors as interactive objects. Even if they did, they were unable to easily find them all. Beyond that, they frequently tried to pull not on the grip area, but on the chain above the grip. As a response, the touch-sensitive area of the switch was changed to also include the chain. Switches were also made larger and a thin white border was added, so that they stand out farther from the background (see Figure 4.2).

Size of switches was increased for better visibility and touch sensitivity.

4.4 Level design

Several specific problems were identified in the initial level design and resolved in a revised version (see Figure 4.3).

Users found it difficult to leave area (1). Creating a bridge at the correct position to guide the pingus up the cliff was too difficult as first task of the game. The cliff was exchanged for a door mechanism (A). The door is the simplest of the

Difficulty of area (1) was decreased.

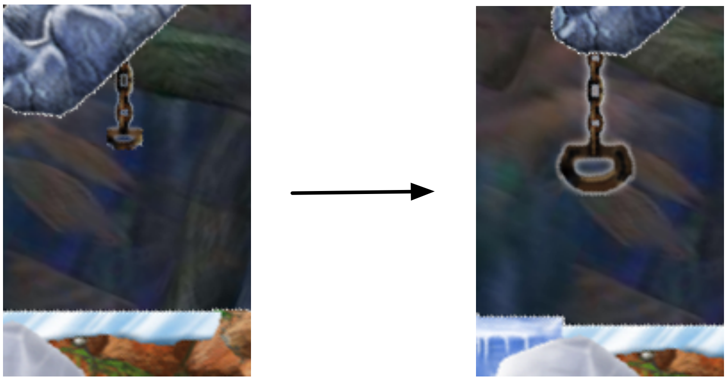


Figure 4.2: Left: Original switch appearance. **Right:** Revised switch appearance.

three gestures, and also has a benign failure mode: when the user inadvertently aborts the gesture, the door simply closes, and the pingus stay in the safe area.

The cooperative aspect was further pronounced.

To further the cooperative aspect of the game, areas (2), (3), and (4) were changed to need even more user actions.

To lead pingus towards area (3), three gestures need to be performed: the door (A) has to be opened, the gap has to bridged or hopped over, and the hill has to be bridged or hopped over. Similarly, in area (3), two bridges are needed for the pingus to bridge the ice pit, and another door (B) has to be opened for the pingus to not turn around and walk into the pit.

Difficulty of area (5) was increased.

Area (5) was very unchallenging in the first design. To increase difficulty, it was changed to always need a bridge for the pingus to be able to proceed towards the door (C). Area (6) was left unchanged.

Although it might seem that the changes increased difficulty because of a stronger need for cooperation, the individual tasks were designed to be easier to perform correctly. User trials revealed no problems.

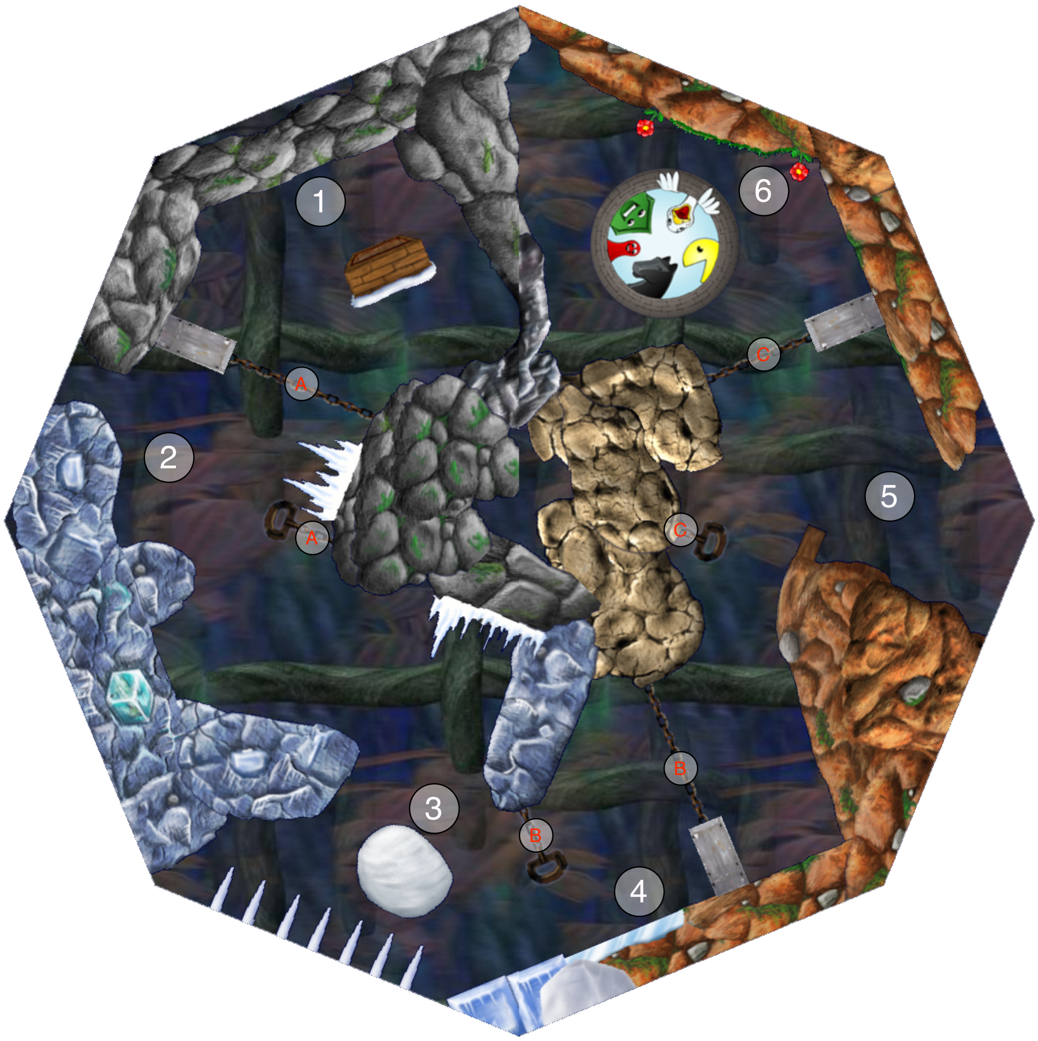


Figure 4.3: Revised level layout with description of areas.

4.5 Score

During user interviews, the lack of any score and performance feedback was criticized. Especially users who were playing multiple rounds asked for comparison information. Reward for winning is part of any game, especially since gratification cannot come from success over other players in this case. Thus, the lack of even a simple reward was a considerable omission in the first design.

Score display and fireworks was added.

To correct this, a score showing the amount of rescued pingus at the end of the level was introduced. It would rotate around the center of the table for equal visibility. Additionally, when more than one third of the pingus were saved, a small firework animation would play (see Figure 4.4).

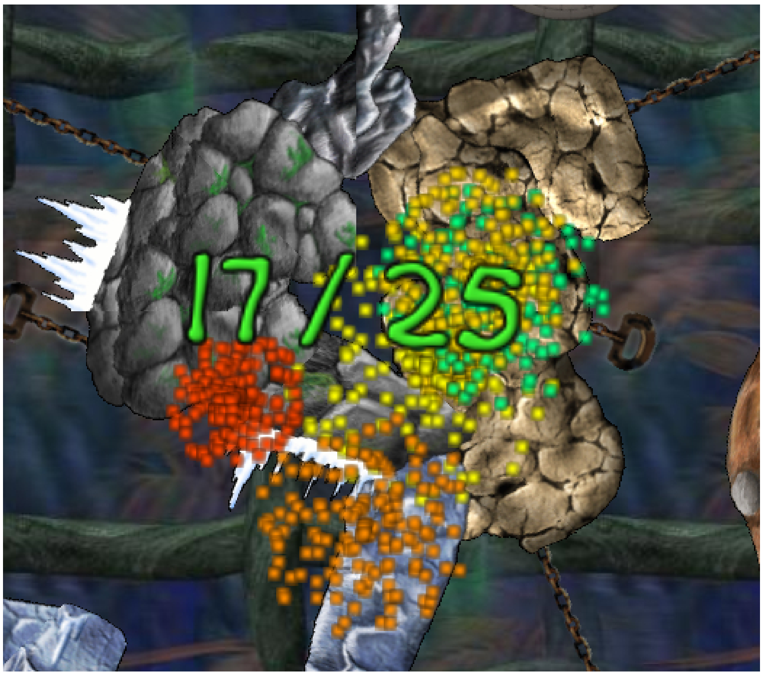


Figure 4.4: Score and fireworks display upon finishing the level.

Chapter 5

Implementation details

"Programming is not a zero-sum game. Teaching something to a fellow programmer doesn't take it away from you. I'm happy to share what I can, because I'm in it for the love of programming."

—John Carmack, id Software

The task of implementing PinguTouch was logically separated into four different areas: adapting the appearance of the game, integrating a touch-detection library, building interfaces to implement gestures in the game, and implementing the actual gestures.

5.1 Appearance

5.1.1 Architectural overview

Pingus uses a framework called "ClanLib"[1] to abstract hardware access. It allows rapid development of 2D sprite-based games and simplifies cross-platform development with its support for managing hardware and software resources, sound mixing, and GUI generation. Most impor-

Pingus builds on ClanLib, which itself uses OpenGL for graphics operations.

[1] See http://www.clanlib.org/.

tantly for modifications regarding the output of the implementation, it also provides abstractions for OpenGL-based rendering. For better understanding of the necessary changes, an overview over the rendering pipeline of the game is given (see Figure 5.1).

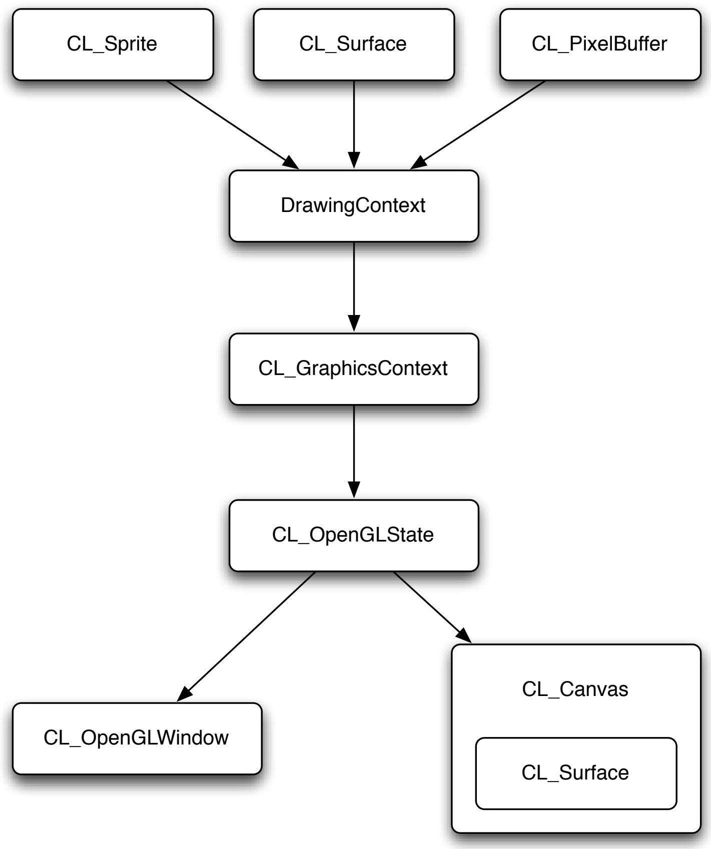


Figure 5.1: Architectural overview of the graphics pipeline; API users can also get hold of `CL_OpenGLState` and use thinly wrapped OpenGL functions directly.

`CL_GraphicContext` is assigned to each drawable surface.

At the lowest level of the ClanLib graphics abstraction lies the class `CL_GraphicContext`. Each drawable surface, i.e., the backbuffer of the display or drawable textures, has an instance of `CL_GraphicContext` associated with it that is retrievable at any time from that surface. This class offers functionality that allows to draw pixels, lines, and triangles, and allows the API user to specify transformation properties for translation, rotatation, and scaling.

Should the programmer need access to a specific OpenGL-feature that is not wrapped by the `CL_GraphicContext`, there is a way to directly work with OpenGL by obtaining an `CL_OpenGLState` object from the `CL_GraphicContext`. Then, the programmer can access normal OpenGL functions by using one-to-one ClanLib-pendants. This has the advantage that ClanLib can take care of saving and restoring the correct state when there are multiple such objects in use. However, drawing sprites on such a low level is tedious work, so ClanLib provides three additional classes with higher level abstractions.

A texture wrapped into a `CL_PixelBuffer` object resides in the main memory of the computer. This has the advantage that it can be processed by the CPU, e.g., as part of collision detection. However, if it is drawn by ClanLib, it first has to be uploaded to the memory of the GPU (graphics processor unit). This is a relatively expensive process performance-wise, that should not be done on a per-frame basis.

Textures of `CL_PixelBuffer` instances reside in main memory, while textures of `CL_Surface` and `CL_Sprite` instances reside in the GPU local memory.

A texture wrapped into a `CL_Surface` object represents a simple texture in the memory of the GPU. The API supports functionality like coloring, blending, 2D transformations and non-power-of-two texture sizes, but does not offer any functionality that is not easily accessible by standard OpenGL commands.

Finally, the class `CL_Sprite` offers everything that a `CL_Surface` offers, with additional support for 2D bitmap animations, e.g., multiple frames packed into one texture.

Additionally, for render-to-texture operations, developers can instantiate an object of the class `CL_Canvas` on top of a `CL_Surface`. It provides a `CL_GraphicContext` for drawing into that surface like into the backbuffer. Internally, it is based on the OpenGL PBuffer extension.

`CL_Canvas` can be used for render-to-texture operations.

Pingus itself uses a class `DrawingContext`, that can buffer up drawing requests of surfaces, sprites, fonts, and other graphical primitives. It can be instructed to draw into a

DrawingContext can buffer and z-sort drawing requests, and draw them into a CL_GraphicContext.

CL_GraphicContext. For this, it performs z-sorting on all buffered objects to reduce overdraw, and then draws all buffered requests front to back.

All logical high level components of the standard game view, e.g., the buttons, status bar, and the playfield containing the actual level, are seperated into own classes. Each frame, all UI components are instructed to draw into one common DrawingContext, that then gets drawn into the CL_GraphicContext of the backbuffer of the GPU. For PinguTouch, all conventional user interface elements were removed from the game, so that only the playfield is visible. Accordingly, that is the component this description is focussing on.

Playfield is the UI component that displays the level; it delegates to SceneContext for actual drawing.

The class Playfield is the user interface component responsible for drawing the landscape and all pingus and other dynamic objects that are part of it.

Playfield holds an instance of the class SceneContext. While Playfield is responsible for handling UI input and determining the viewport of the scene, the actual drawing requests are performed by the SceneContext. It holds all information necessary to render the landscape fitting to the position of the viewport and can perform clipping, rotation, and zooming operations.

The landscape gets divided into tiles.

The landscape itself is composed of simple sprites during level creation. These sprites can be flipped horizontally or vertically, and rotated in 90 degree steps to include more variation. Instead of rendering all these sprites each frame one after another, the level is divided into quadratic tiles of the size 32x32 pixels during level startup. Then, all the static graphical artwork of the level is rendered into their respective tiles. For final output only those precomputed tiles are rendered.

This method eliminates overdraw, i.e., for displaying the static playfield, pixels will never be written to more than once during rendering of a frame. This comes at the price of losing the ability to easily remove only certain sprites from a tile, since it is impossible to decompose the tiles back into sprites. No game mechanic needs that ability, however, so this disadvantage has no impact.

During the course of the game, accurate and fast access to collision information is necessary to properly move the pingus along the game world. To achieve constant-time access to the collision information of an arbitrary pixel of the level, the data is held in a simple two-dimensional byte-array of the size of the level. Whenever a sprite is drawn into the tilemap, the corresponding bytes in the collision map are marked. Possible collision markers are empty areas, ground areas, indestructible areas, and water areas.

Collision information is held in an array the size of the level.

5.1.2 Modifications

To create an octagonal output, the level can simply be divided into eight triangles for level editing and put together along the sides of the triangles for display (see Figure 5.2). Since the artwork composing a level is primarily rectangular formed, sprites placed into one triangle could sometimes reach into the next (see Figure 5.3). To alleviate this, a 150 pixel wide gap was introduced between the triangles.

The level is divided into eight triangles that are put together at their edges.

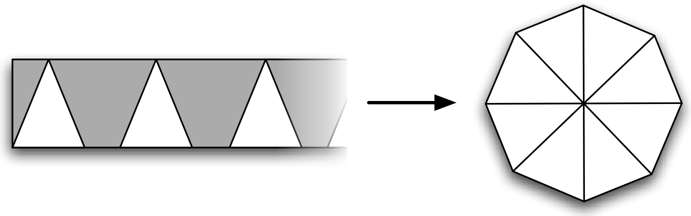


Figure 5.2: Eight triangles (not all visible in this figure) put together to form an octagon.

Since the concept is similar to a fan with blades, the term "blade" will be used from here on to describe these segments. For non-ambiguous description of the output, linear level coordinates will still be used for the description of position. Accordingly, the beginning of a blade denotes the pixel associated with the lower left corner of the corresponding segment in level coordinates.

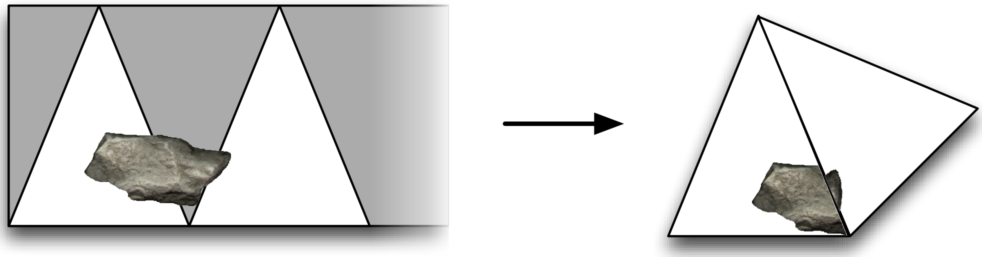


Figure 5.3: Illustration of artwork reaching into neighboring triangle.

When pingus leave one blade, they are moved to the next.

Apart from splitting the level into blades and drawing them in form of the octagon, transitions of pingus between the blades must be handled. From a game logic perspective, this is easily done: the bottom center point of every pingu is used as its logical coordinate. This point is used as origin for drawing and collision detection. Whenever a pingu moves far enough for the origin to leave the visible area of a blade, it is immediately moved to the beginning of the following blade in the direction it is walking.

The visual aspect needs to be handled separately. Even if the logical coordinate of an object is still in one blade, parts of it might be drawn outside of it and will get cut off at the seams of blades. To resolve this behavior, we examine the basic trigonometry of regular polygons.

Obviously, any sprite should be in its original rotation again after crossing through all 8 blades, i.e., it gets rotated by 360 degrees. Since the output gets displayed in a regular octagon (i.e., all edges have equal length, and all inner angles are equal), it needs to be rotated by 45 degrees during each step.

Pingus are drawn on both sides of gaps so they are not clipped.

Thus, when part of an object reaches outside a triangle, a copy of its sprite needs to be moved by 150 pixels to account for the additional gap between triangles, and then rotated by 45 degrees in the appropriate direction around the beginning of the triangles it reaches into (see Figure 5.4).

The main part of the modification was done in the

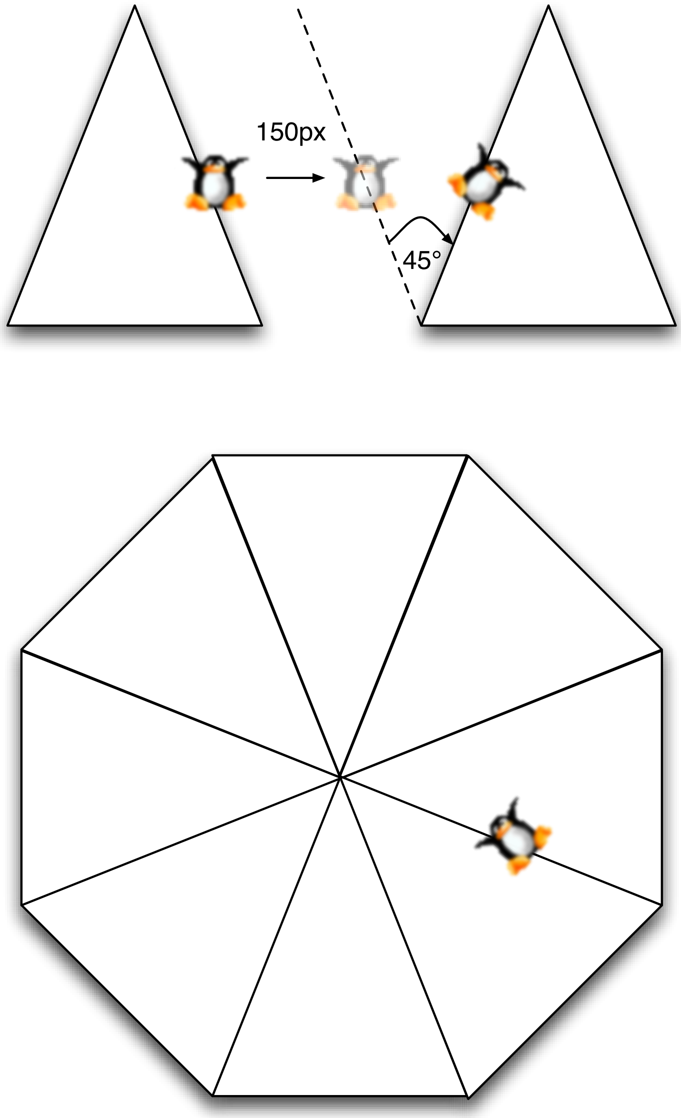


Figure 5.4: Sprites reaching outside one triangle are duplicated, moved, and rotated to appear seamless.

`SceneContext`. The viewport is reduced to the size of the minimal containing rectangle of a blade. Then, for each blade, the viewport is moved to the respective position on the level, and rendered into a texture. Finally, when the game engine requests the `SceneContext` to be drawn, these textures are arranged in the octagonal form.

Each blade is rendered into a texture.

Since the textures themselves are rectangular, as opposed to the triangular blades, some surplus texture data will be discarded. This must be kept in mind when designing the level.

To compute the correct vertex coordinates of each blade, one needs know how to calculate the set of points that make up a circle.

The equation

$$(x, y) = (\cos(\alpha), \sin(\alpha)) \quad | \quad 0 \leqslant \alpha \leqslant 2 \cdot \pi$$

delivers all points on the perimeter of the unit circle. To obtain the vertex coordinates of the 8 points, α has to be restricted to the values of

$$\alpha_i = \frac{\pi}{4} \cdot i | i \in \{0, \ldots, 7\}.$$

The background image is drawn as a seperate layer to make it appear as a single piece.

The background image is a special case. Although it is part of the landscape and as such should be painted as part of the `SceneContext`, the segmentation along the edges of blades would lead to highly visible seams. To prevent this, it is instead drawn in a seperate pass.

To make it fit into the octagon form of the level, a stencil buffer is used: in a pre-pass, the stencil buffer is populated in an octagonal form, using the same geometry that the drawing of blades uses. The background is then only drawn on pixels that pass the stencil buffer test.

5.1.3 Optimizations

In the first implementation of PinguTouch, the whole scene was rerendered for each frame. The pingus and all other dynamic objects were kept as part of the `SceneContext`, like they were in the original Pingus design. This kept the rendering pipeline simple, as all objects were automatically drawn at the correct position with the correct rotation.

However, rendering the whole level into different textures and then placing those on the screen is quite an overhead. To improve performance, the architecture was changed so that the rendering output of the first frame gets cached. Then, only the dynamic objects get drawn on top of that. This added some complication to the codebase: all dynamic objects now had to be rotated and placed independently into the `DrawingContext`.

Static parts of the level are cached, dynamic parts are drawn on top for better performance.

For that, a transformation from level coordinates to screen coordinates for arbitrary points is needed. The following algorithm was used:

1. The blade which contains the point is computed. For this test, ClanLib provides functions to check whether a point lies within any specific triangle. Should the point be outside of any blade, it is not visible on screen, and further calculations can be skipped.
2. The vertical distance d_v between the lower edge of the blade and the point, and the horizontal distance d_h between the left edge of the blade and the point is measured.
3. To compute the output position, the lower left point of the blade position on screen is used as starting position. From there, it is moved by d_v pixels along the height axis that is passing through the center of the circular output, and by d_h pixels parallel to the lower edge of the blade.

The reverse transformation from screen coordinates to level coordinates works similarly:

1. The function calculates the on-screen blade the touch occured in by a simple bounds check of the touch point for all blades. The level coordinates of the beginning of the resulting blade serve as a starting point for the following computations.
2. The offset to the x coordinate is obtained by measuring the distance from the left edge of the on-screen blade to the touch point along an axis parallel to the lower edge of the blade.

3. The offset to the y coordinate is obtained by calculating the distance from the center of the circular output to the touch point.

5.2 TouchLib

As low-level framework for integrating multi-touch functionality into the game, the "TouchLib" library[2] was used. TouchLib provides the embedder with solid calibration and cascading image filter possibilities for usage on an FTIR setup (cf. Section 2.2.4—"Frustrated total internal reflection") to obtain touch point information.

TouchLib provides calibration and cascading image filters to obtain touch point information.

Next to position and identification across frames, parameters describing an ellipsis approximating the blob are supplied for each touch point. However, early in the development process it became apparent the performance was insufficient for the purposes of this implementation. For assigning the identifiers to touch points from one frame to the next, the library uses the following exponential time algorithm:

1. The distances between all touchpoints from frame F_{n-1} and all touchpoints from frame F_n are calculated
2. All possible permutations of fitting the points from F_{n-1} to the points from F_n are tested
3. Keep the permutation where the sum of the distances of the assigned points is minimal.

TouchLib performance was improved.

TouchLib was modified to replace this algorithm with a more pragmatic and faster one:

1. Again, the distances between all points from F_{n-1} and F_n are calculated in a matrix.

[2] See http://www.whitenoiseaudio.com/touchlib/.

2. The assignment with the lowest distance is chosen (i.e., the smallest value in the matrix), and the distances involving these two points from F_{n-1} and F_n are marked as processed and removed from the matrix.

3. Step 2 is iterated until no unassigned points from either F_{n-1} or F_n are left.

This very simple algorithm runs in $\mathcal{O}(n^3)$, and has not shown performance problems in any of our test scenarios. Figure 5.5 displays an example where the output of the two algorithms do not match. However, there has yet to occur a case during our user studies where such a mistake would have been apparent.

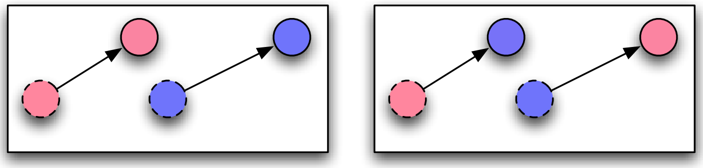


Figure 5.5: A case where the two identifier assignment algorithms lead to different results. Arrows represent movement paths, colors represent identification. **Left:** Identifier assignment when minimizing global distance. **Right:** Identifier assignment when picking local minimal distances.

Capture resolution sufficed, but capture rate raised issues.

The capture resolution of the camera input used in this project is 320x240 pixels at 30 frames per second. With a game surface of about 85 centimeters, the system can thus differentiate between single points of ~ 0.35 cm distance. However, since the coordinates used as touch points are based on blobs that are larger than single points, the actual quantization of the returned data is somewhat finer grained, as the system can interpolate the center of the blob based on its dimensions. During tests and deployment of the system, there were no apparent cases where lack of resolution seemed to impair input control.

The system would definitely benefit from a faster capture rate, however. At the current rate, motion blur effects can lead to distorted input data, that has to be factored in when developing the software (cf. Section 5.4.3—"Blocking and bridging").

5.2.1 Calibration

Calibration procedure needs the user to touch predefined points.

The mapping of the blobs to actual coordinates is implemented in TouchLib by reverse texture mapping.[3] During the calibration process, a regular grid of points is displayed. The user then has to touch the screen at each point, so that the software can derive the coordinates from that specific spot. Internally, the calibration software segments the entire work area into triangles, using neighboring points as vertices. When a blob is detected, TouchLib tests which triangle contains it, and then uses the respective parameters for the transformation.

Calibration grid was changed from rectangular to octagonal.

Since the evaluation took place on an octagonal shaped table, the integrated calibration routine had to be adapted. Instead of a regular grid of points, two concentric octagonal shapes were used (see Figure 5.6).

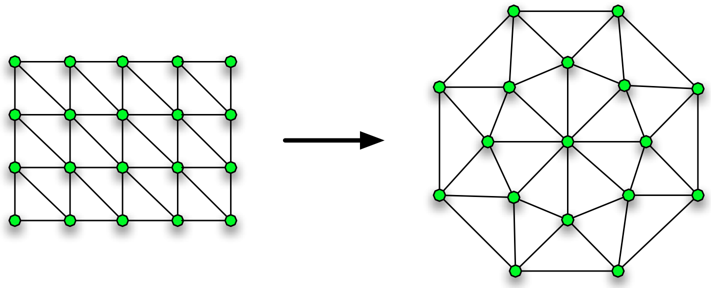


Figure 5.6: Rectangular Calibration Grid and Octagonal Calibration Grid; the segmentation into triangles is not visible to the user.

[3] See http://www.cescg.org/CESCG97/olearnik/txmap.htm.

5.3 Interaction

The touch data is fed into a graph.

For usage of the touch data in PinguTouch, the raw identification and position information of the touch points is fed into a graph structure. The vertices of that graph represent the touch points, while the edges hold the distances between the points. From there on, all gesture recognition is performed on the graph structure as visualized in Figure 5.7.

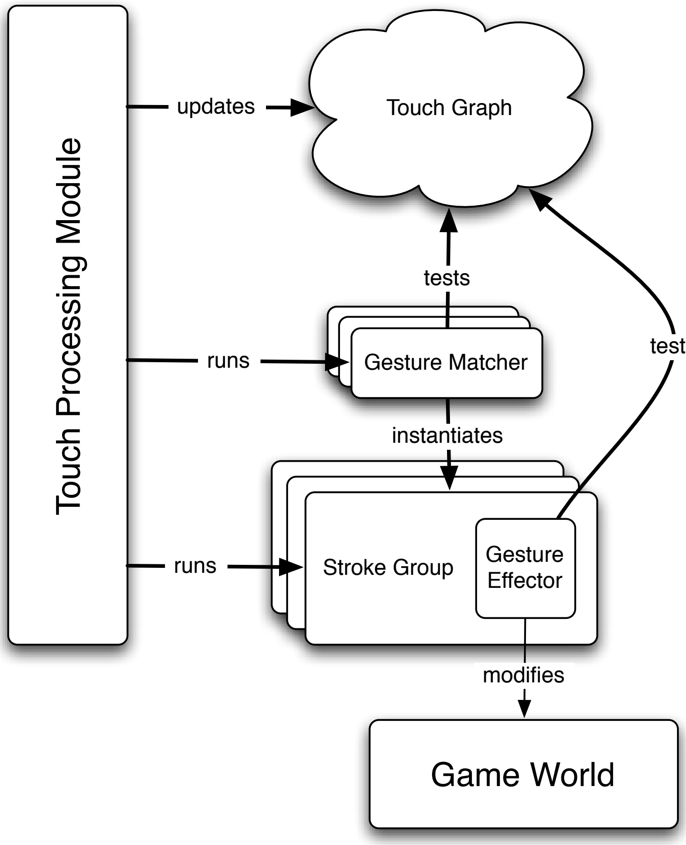


Figure 5.7: Overview of the touch system implementation.

For implementation of the graph, an adjacency list from the Boost Graph Library[4] was used. It was chosen over the

[4] See http://www.boost.org/libs/graph.

adjaceny matrix from the same library because references to vertices are memory pointers and stay valid even after items were added to or removed from the graph. This is an important technical advantage.

For integrating gestures, two interfaces have been developed (cf. Appendix B—"Interface definitions").

Implement `IGestureMatcher` to describe gesture match conditions.

For the recognition process, `IGestureMatcher` has to be subclassed and registered with the touch processing module. The order in which the gesture matchers are registered determines the priority of the gestures.

For processing of a recognized gesture, `IGestureEffector` has to be subclassed.

Any subclass of `IGestureMatcher` needs to implement only two functions: `checkForMatches` receives the current graph structure as argument and generates a list of `StrokeGroup`s, each containing a set of touchpoints recognized as matching a new gesture. `generateEffector` receives a vector of graph vertices and returns an initialized instance of the respective implementation of `IGestureEffector` that belongs to the gesture matcher.

When an instance of the class `StrokeGroup` gets created during successful matching, it receives the instance of the gesture matcher and the touch points, that are detected as part of the gesture, as arguments. The `StrokeGroup` instance then lets the gesture matcher generate an effector with the members of the gesture.

`StrokeGroup` handles common functionality.

`StrokeGroup` instances only serve as wrappers around multiple touch points. They handle the functionality common to all gestures, i.e., grouping the touch points, memory management and logging functions. Actual game play changes are performed by the effectors.

A list of active `StrokeGroup`s is maintained. During each frame, this list is traversed and each `StrokeGroup` instructs its effector to perform its action by calling `perfomEffect`.

During that time, the effector can signal with its return value that it is not required anymore, usually when some of the touch points belonging to the gesture are removed. If that is the case, the StrokeGroup and the effector are destroyed. All remaining touch points that were part of the gesture are declared free for usage in new gestures.

IGestureEffector can decide when to end gesture.

The functions getGestureID and getGesturePos can be used for logging purposed and respecetively return a unique identifier for the type of gesture the effector implements, and the position the effect occurs at.

Two deliberate design decisions were made to simplify implementation:

Strokes belong to at most one gesture and cannot be added to an existing StrokeGroup.

Each stroke belongs to at most one gesture effector. This should be no restriction in most practical cases.

After a gesture effector has been created, no additional strokes can be added to it. While this restriction might limit the variety and stability of gestures, it also leads to more predictable behavior for the user: gestures performed in high vicinity cannot interfere with each other, as long as the respective gesture effectors are not created at the same time. For example, one player can create a blocking gesture, and another player can then perform flicking gestures nearby without the possibility of them getting recognized as part of the blocking gesture.

5.4 Gestures

5.4.1 Flicking of pingus

The gesture matcher checks each stroke that is not in use by another gesture for two criteria:

Flick gesture criteria involve stroke duration and proportions.

The stroke must be shorter than 2 seconds. This criterion disallows triggering the gesture when users are simply resting a finger on the surface and a pingu walks into it. It still allows for enough time to move a finger from one side of a pingu, across the pingu, to the other side to prod it towards that direction.

The width-to-length ratio of the stroke must be smaller than 1.8. This restriction is designed so that the gesture only gets triggered when the tip of the finger is used. Specifically, the edge of the hand that is used to trigger a blocking gesture usually has a higher ratio, so it will never trigger a flicking gesture. The exact value of 1.8 has been acquired empirically.

After successful matching, the user has to move the finger 30 pixels from the current location for the pingu to get propelled into that direction.

The path of flying pingus was changed to be continuous in screen coordinates.

In the first implementation, the flight path of a flicked pingu was continuous according to level coordinates, apart from moving the pingu over the level gap from one blade to the next. User tests showed this to be very confusing when shown on the screen, as the pingu made a sharp turn when crossing between two blades (see Figure 5.8).

The behavior was changed so that pingus, that were not in contact with the ground, were moving in a straight path on-screen. This was achieved by rotating the pingu's velocity vector by 45 degree in the proper direction whenever it crosses from one blade to the next (see Figure 5.9).

5.4.2 Opening doors

All world objects are queried whether to react to touch.

To implement the opening of doors, a general solution for interaction with world objects was designed. For each touchpoint, all world objects are iterated and queried whether they want to react to that specific point by calling `fingerDown` with the level coordinates of the touch point. If so, all further stroke information for that point is then routed to that object by calling `fingerUpdate`. The world object can control with a return value whether it keeps control of the stroke. If it does not, `fingerUp` gets called to handle any necessary clean-up operations.

A severe limitation of the current design is that world objects get only fed one stroke at a time, so that gestures con-

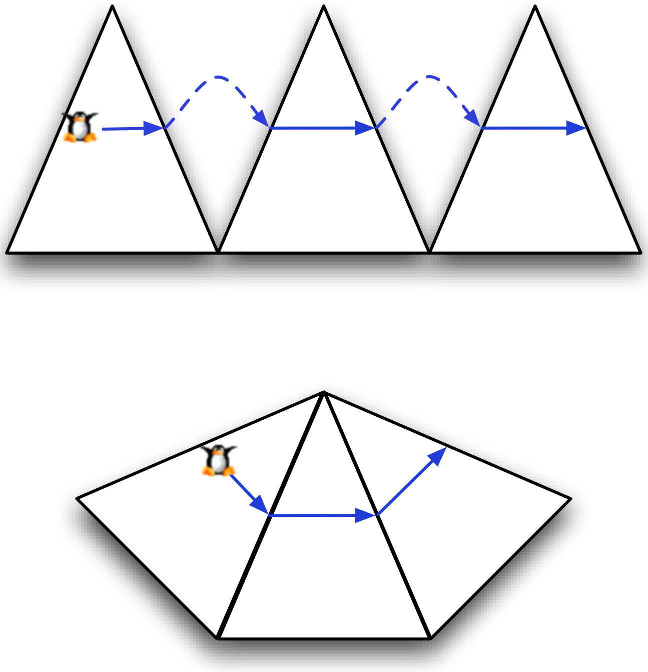


Figure 5.8: Traversing of the pingu between blades with continuous level coordinates.

sisting of multiple strokes are very hard to implement with this general solution. This is sufficient for the current usage, i.e., for pulling levers, but might need a redesign in possible future versions.

Current design supports only one stroke per world object.

To get attached to the door in this specific case, the code checks whether the stroke hits the lever of the door. Then, when the player moves the finger, the lever gets pulled with it along the vertical axis. The horizontal movement is ignored. Should the player lift the finger, the lever slowly moves back into its original position.

During each frame, the collision map is updated. The position that the door was placed in the previous frame is cleared in the collision map and the new position is marked as solid. The code assumes that the door does not clip the ground of the level in its closed state. If it would, this intersection would be marked as clear in the collision map when

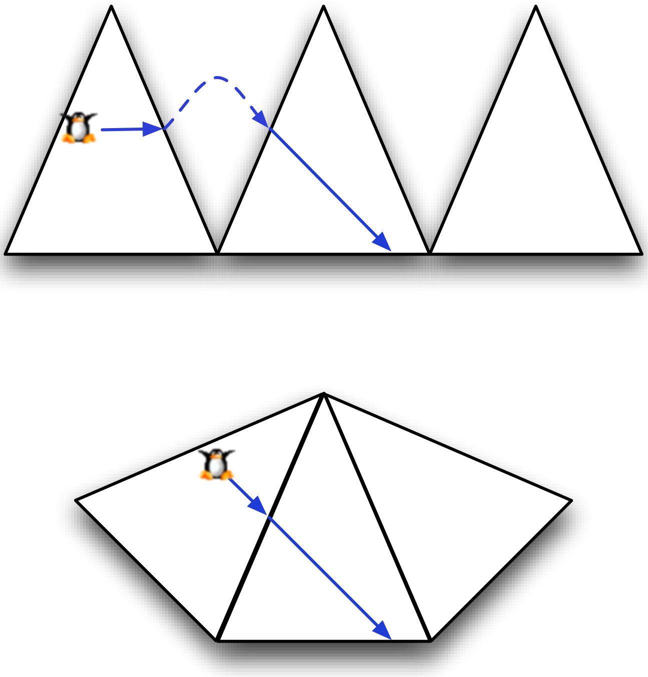


Figure 5.9: Traversing of the pingu between blades with continuous screen coordinates.

the door is open, even when the ground is still there.

5.4.3 Blocking and bridging

Players have differently shaped hands.

Different players have differently shaped hands or place their hand on the table in different ways. With some players, only one blob is visible, reflecting the edge of the hand (excluding the pinky finger). With other people, an additional blob is detected for the pinky finger (see Figure 5.10).

To accurately detect which blobs are part of the hand gesture, the following algorithm is performed:

Block gesture criteria involve proportions and stability.

The list of strokes is searched for a seed from which to perform further searches. It has to fulfill the following criteria:

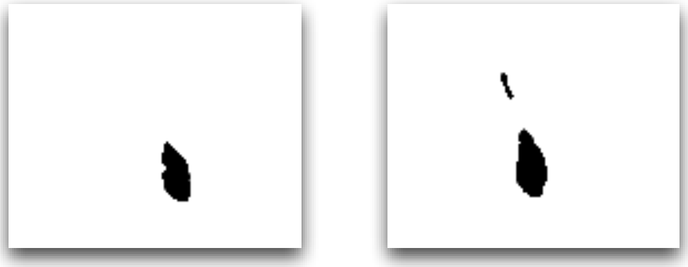


Figure 5.10: Possible blob configurations detected by placing the edge of the hand on the table. Image colors are inverted for clarity.

The ratio between length and width of the point has to be at least 2.2, and the length of the point has to be at least 4 pixels in terms of the camera input. The first value ensures that the point is long enough to actually represent the edge of a hand. The second value enforces that very small points, which can reach the necessary ratio quite easily simply out of chance, are ignored. These values were gathered empirically for children. For adults, a length to width ratio of 3.0 has led to good results.

The point needs to have stable coordinates for at least 3 camera frames. This criteria is only for technical reasons. When a player quickly moves his finger across the surface, the resulting motion blur in the picture makes the blob appear larger than the size of the contact actually is. The timeout reduces false positives due to this effect.

At a camera input of 30 frames per second, this restriction introduces a lag of 100 milliseconds from the time of placing the hand until the bridge actually appears. As explained by Card et al. [1983], this latency is just at the threshold needed for real-time interaction and thus should not measurably impact the user experience.

When a seed is found, all touch points in a diameter of 150 screen pixels from the seed are tested for a similar axis and a similar rotation as the seed. If any are found, they are added to the gesture.

Nearby touch points with a similar axis are added to the gesture.

Then, the mean of the coordinates of the seed and all touch points added in this way is used as center coordinate for the block. If no additional point is found apart from the seed, its coordinates are used. In this case, the block might not appear at the exact spot the user expects: it is shifted somewhat towards direction of the arm. Since it is not reliably possible to detect the direction of the hand that creates the touch points, the software cannot correct this shift.

For drawing, a texture wrapped in a `CL_Surface` is used. This texture is drawn directly on top of the playfield, no regard has to be taken whether it crosses between blades. It can use the center coordinates and the orientation directly from the associated touch point.

Block collision data is rotated and drawn into neighboring blades.

For entry into the collision map however, the touch data has to be transformed to level coordinates. All this is handled on the CPU, so the texture is duplicated in a `CL_PixelBuffer`. The special case of the block reaching from one blade to the next must also be handled.

For that purpose, each block-effector holds three `CL_PixelBuffer`s. One is rotated the same way the drawn texture is, relative to the orientation of the blade it is drawn into. The other two are rotated 45 degrees clockwise and counter-clockwise relative to that, respectively. The collision data is then drawn into the collision map similar to the way used for objectings spanning over multiple triangles (see Figure 5.11).

In contrast to the collision map of the original game, PinguTouch needs the ability to selectively remove elements not only on a per pixel basis, but on a per sprite basis. This is necessary when a blocking sprite that partly overlaps the ground map gets removed from the level.

The additional touch collision map is rebuilt each frame.

The solution used here is that parallel to the normal level collision map, a touch collision map is constructed with the same dimensions. At the beginning of each frame, this map is cleared, and the data for each block is drawn into it. Then, when testing for possible collisions, the union of both maps is checked.

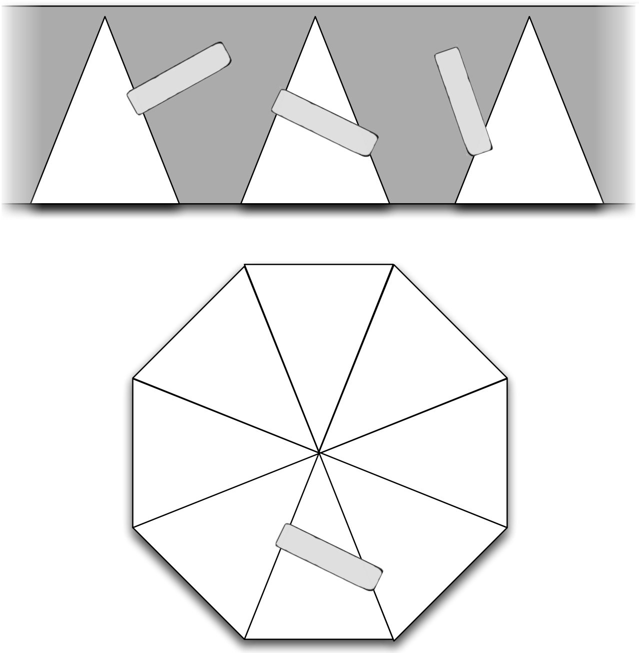


Figure 5.11: Block placement in collision map to appear as one piece.

5.5 Level editor

As part of the original Pingus game, a level editor is included. Unfortunately, the level editor for Pingus version 0.7, the version used as a basis for PinguTouch, was not in a usable state at the time of writing. However, a version included in the version 0.6 of Pingus could be used. The file format used to save the level has changed between versions, but since both are based on XML, an easy transformation between the two formats was possible by usage of an XSLT-transformation. An according stylesheet is included in Pingus 0.7. Although the multi-step process between designing a level and actually playing it was hindering rapid development, it was adequate for development of the level used for further evaluation as part of this thesis.

The level editor from a previous version had to be used.

To better support the creation of levels appropriate for usage in PinguTouch, the editor was modified to highlight

Areas visible in-game were highlighted.

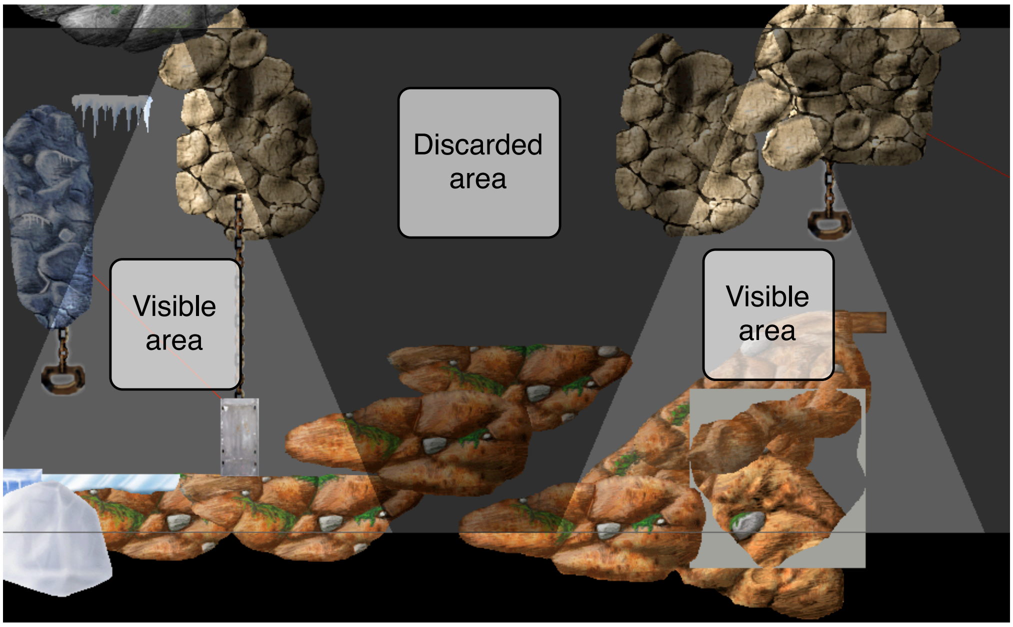


Figure 5.12: Example screen of the level editor. The areas denoted with a dark grey background will be discarded for gameplay. Only the light gray areas will be visible.

the areas that would actually be visible during the game. This was done using a simple background layer (see Figure 5.12).

Another added ability is proper handling of level pieces that extend outside one blade and into the next. For this to render without visible seams in the final output, the groundpiece has to be rotated and moved to the correct place in the world. Since the version of Pingus that the editor is based on uses an older version of the ClanLib framework, implementing proper visualization in the editor for this feature proved to be too much work for only minor convenience during level editing.

Level editing by image manipulation was integrated.

As an additional method for creating levels, a single image file containing the landscape as it would be displayed in the game can be used. This enables level editing by using an image manipulation application. Dynamic objects like entrances and exits still have to be placed in the editor, however.

Chapter 6

Evaluation

> *"When I show a game to people I don't ask their opinion or give them a survey. I just watch their eyes and their face while they play. Do they smile? Do they look frustrated? So I guess I do test my games—but it isn't very scientific."*
>
> *—Shigeru Miyamoto, in an interview with EW.com*

Cooperation has to emerge from the players' actions.

To measure whether we succeeded in meeting our design goals of PinguTouch, we planned and performed user evaluations. Complexity of gestures and real-timeness of the system are properties of the application that can be explicitly set during the design phase. However, cooperation, and thus, perceptibility of co-location, has to emerge from the players' actions.

Evaluation of entertainment value is hard.

Since the primary purpose of our game is entertainment, a study regarding enjoyment was also necessary. Evaluating the entertainment value of an application requires different methods than evaluating the performance of task-based applications. Enjoyment is hard to measure quantitatively, and individual preferences of players can heavily influence the results. While we investigated creating different versions of PinguTouch to specifically isolate how cooperation changes assessment of the game, it has proven difficult to remove single rules of the game without completely destroying the gameplay.

Singling out individual factors was deemed impractical.

Removing the quasi-modal property of the gestures, so that users only have to remove the obstacles once and let the game play by itself, would for all intents and purposes remove the real-time aspect of the game. Since all tasks necessary to win the game are conceptually easy to recognize, the lack of feeling achievement would reduce enjoyment. Similarly, it seems obvious to us that playing the game alone leads to a less entertaining experience than playing in a group.

Two-part study for analyzing cooperation and entertainment with visitors at the Industrion.

To confirm the viability of multi-touch gaming and of our design, we eventually decided to split the evaluation into two parts. In the first part we analyzed whether users played PinguTouch as intended, i.e., using the co-located setup for close cooperation. The second part was designed to confirm whether users actually enjoyed the game.

Advantages: diverse demographics, octagonal table, gaming atmosphere.

Both parts were performed with visitors at the Industrion museum during normal opening hours. This had several advantages over performing a study in a controlled environment in our lab:

- The demographics at the museum are much more diverse than any that we could have realistically procured from the university campus.
- The multi-touch system at the Industrion features an octagonal table, fitting to the design of the game. The table in our lab, on the other hand, is rectangular, thus impeding accessibility.
- We felt that the work-oriented atmosphere of a lab might influence the results compared to an environment more suitable for playing games. Discrepancies of that kind have been discovered in [Reilly and Inkpen, 2007].

Disadvantages: limited data gathering, no fixed testing procedure, mostly families.

On the other hand, conducting user studies in the uncontrolled environment of the museum had several repercussions:

- Our methods of gathering data were generally limited to only the most unintrusive ways. Visitors go to the museum for education and entertainment. It is unreasonable to expose them to the potentially intrusive process of an in-depth user study.

- Visitors would come and go as they pleased. There was no mechanism to enforce specific tasks. Although we gave a short introduction to users who approached the table, there was no clear instruction or testing phase. After quickly explaining the goal of the game and the three possible gestures, users were told to proceed as they like.
- Museum groups are often families. Family members have a tight bond, and their behavior might not be transferrable to groups in general.

6.1 Part one - analyzing cooperation

6.1.1 Methods

Video observation was used to measure cooperation.

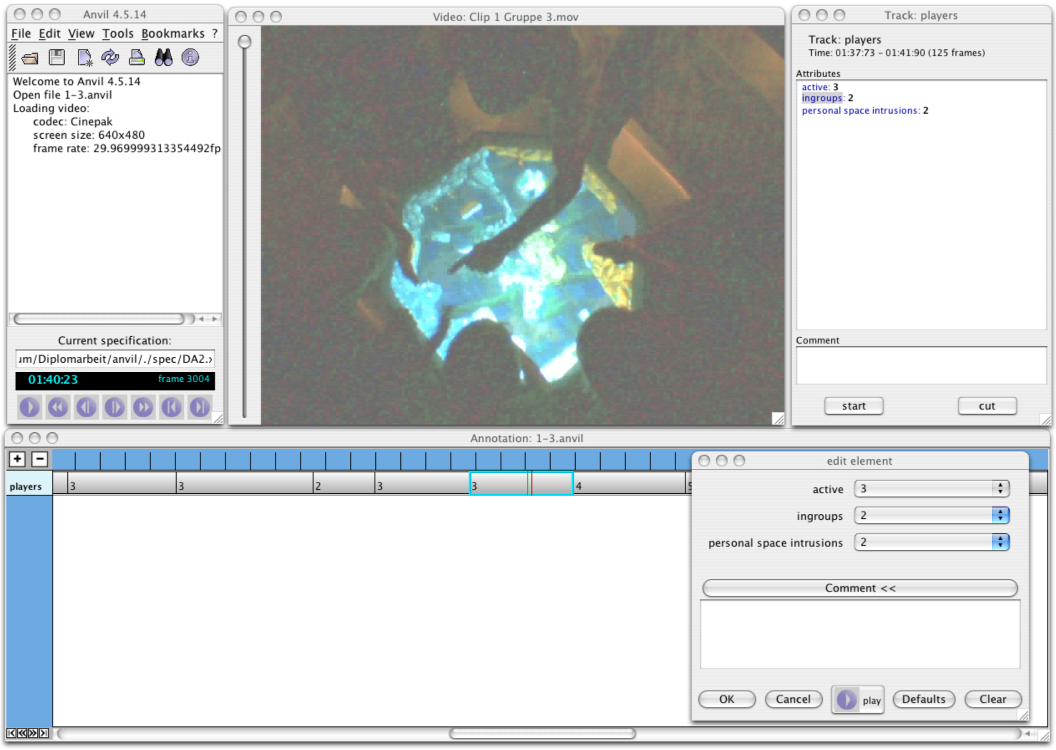
To examine the degree that users were working together, video observation was used. A video camera was mounted above the multi-touch table and set to record the happenings on the table over the course of a day. The camera was carefully set up in a way that users were not identifiable on the video feed.

As the first step of our analysis stage, we segmented the video data into clips containing one group each. Since players could join or leave the game as they pleased, the group size did not necessarily stay constant during that time.

Video annotation by number of users, users working together, users working in personal space of others.

The tool "Anvil" by Kipp [2001] was used to annotate the video clips along a timeline (see Figure 6.1). We noted three pieces of information for each point in time:

- **the number of active users**, i.e., users that were performing a gesture on the table with the purpose of advancing the game, as opposed to simply experimenting



```
Welcome to Anvil 4.5.14
Open file 1-3.anvil
Loading video:
    codec: Cinepak
    screen size: 640x480
    frame rate: 29.969999313354492fp
```

Current specification: ım/Diplomarbeit/anvil/./spec/DA2.›

01:40:23 frame 3004

Track: players
Time: 01:37:73 - 01:41:90 (125 frames)

Attributes
active: 3
ingroups: 2
personal space intrusions: 2

Comment

active: 3
ingroups: 2
personal space intrusions: 2

Figure 6.1: The Anvil workspace.

- **the number of users sharing a workspace**, i.e., users that were working in the same segment of the octagonal playfield
- **the number of users working in the personal space of another user** as defined in Section 2.1.1—"Territoriality in collaborative tabletop workspaces"

The focus on cooperation might lead to decreased importance of personal space.

The third case was particularly interesting to us, since the intrusion of one player upon another player's personal space is a situation that would usually not occur. An investigation on whether the focus on cooperation would lead to a significant decrease of territoriality regarding personal space might lead to further insights about tabletop behavior. The difference between the general case of cooperation and the specific case of working in another user's personal space is illustrated in Figure 6.2.

From the annotated information we could derive three new

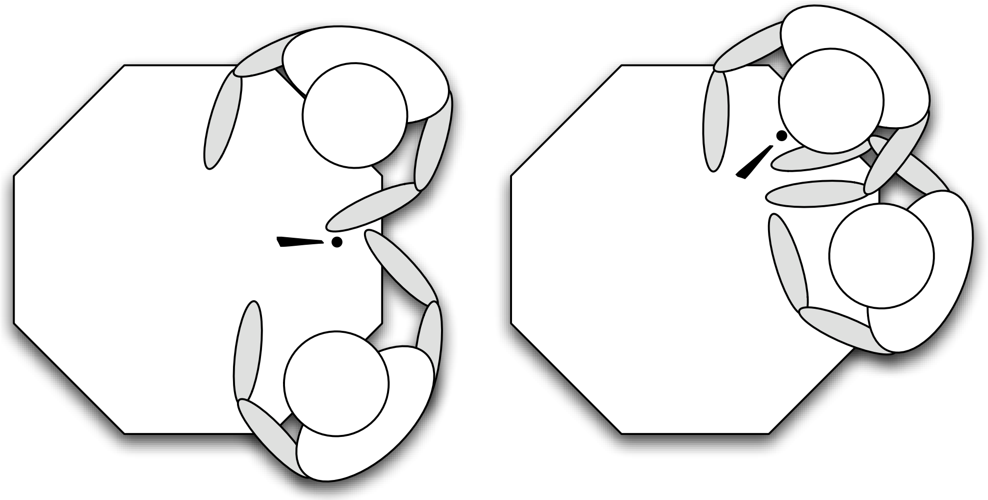


Figure 6.2: Possible ways of cooperation. **Left:** simultaneous activity in a space between players. **Right:** simultaneous activity in a space in front of a player.

values:

- **Summed active play time** sums up the active time of all respective players.
- **Cooperation participation rate** divides the summed up time of users sharing a workspace by the summed active play time.
- **Personal space intrusion rate** divides the summed up time of users intruding into another users personal space by the summed active play time.

We derived summed active play time, cooperation rate, intrusion rate.

A typical situation would involve three players, with two working in the same area, and a third handling another area of the game. The cooperation participation rate in this scenario would be 66%.

Since user identification was impossible, the figures could not be broken down into time per individual, as was done in [Scott et al., 2004].

No user identification possible.

6.1.2 Results

We reviewed 40:29 minutes of video, separated into 7 groups (see Table 6.1). Average cooperation participation rate was 64%. Average personal space intrusion rate was 21%.

Game also playable by larger groups.

We observed two more groups that were too large for meaningful analysis. There was equal activity in all all areas of the table, and no fixed personal areas could be defined. Although we did not include these results in our calculations, we feel that the fact that groups of that size could successfully play PinguTouch confirms the viability of multi-touch gaming.

6.1.3 Discussion

Complexity of PinguTouch right for museum visitors.

For the museum visitors, the complexity of PinguTouch seemed to find the right balance. Users were engaged and entertained by the game for up to 9 minutes before moving on.

We cannot isolate single factors in our comparisons.

To assess whether we succeeded in creating a cooperative game, we compare PinguTouch to the system described in Section 2.1.2—"Collaborative Coupling". However, we cannot isolate the difference in results to one specific factor, since the methodologies between the two studies are different:

Trials in [Tang et al., 2006] were always performed with pairs of users, while PinguTouch was often played by larger user groups. Especially for odd-numbered group sizes, limited reachability hinders cooperation involving all participants for purely physiological reasons. Furthermore, users in the study by Tang et al. were using styluses to interact with the table. This increases the social distance when compared to interacting using the hand directly. Design of the table was different as well: while the table used for the routing application was rectangular, ours is octagonal.

Despite the differences, we can still come to conclusions re-

Group	Play time	Summed active play time	Cooperation paticipation rate	Personal space intrusion rate
1	7:46	17:55	54%	21%
2	6:37	15:21	70%	22%
3	3:45	9:54	72%	36%
4	8:44	17:50	52%	14%
5	3:11	5:51	83%	0%
6	3:10	5:35	81%	32%
7	7:16	14:53	64%	22%
Avg / time			64%	21%

Table 6.1: Video analysis results.

garding the system as a whole, including the variation in both the respective applications and table setups.

Tang et al. identify six different coupling styles. The only one associated with our concept of cooperation is "Same problem same area": both users are working on the same area to cooperatively solve their task. Pairs spent 33,2% of their time in this work style, compared to 64% in our application.

Our system compares very favorable regarding cooperation.

Since the study by Tang et al. was also designed with cooperation in mind, our results show that we succeeded in our goal to encourage close spatial cooperation—at least compared to another, similar application. This could either be due to the application itself or due to another, external factor as part of the whole system.

As an additional step, we analyze our data regarding the further going concept of personal space intrusion as discussed in Section 2.1.1—"Territoriality in collaborative tabletop workspaces".

Group 5 had a 0% personal intrusion rate.

In this context, the result of group 5 is particularly noticeable. It is the only group with 0% personal space intrusion rate, that at the same time has the highest measured cooperation participation rate. An analysis of the video record of that group showed that it consisted of only two players. One player stood at a constant position, while the other player moved around the table helping the first player to handle the pingus to either his left or right side. There was never a situation where intrusion into the respectively other player's personal area was necessary. As this is a valid way to play the game as designed, we kept the results of those groups included in our data.

Many differences in methodology.

Variations in methodology are more pronounced than in the previous comparison. Again, the table design is different: the table used by Scott et al. was round and all areas were equally usable. A large part of it was covered with a cardboard floor plan, implicitly designated as group space. On the other hand, the table edges of our octagonal table are not usable, due to the design of both the table itself and the game level. The area of interaction was generally shifted towards the center. Users in the study by Scott et al. were sitting mostly in the same position relative to the table, while users playing PinguTouch were standing and had to regularly change positions to properly play the game.

Different unit of measurement.

All these differences could again be considered part of the respective systems and included as part of the comparison. However, the unit of measurement was also different: [Scott et al., 2004] evaluated personal space intrusions per actions, whereas we measured them by time spent in that space. To make the numbers vaguely comparable, we will assume that each user action performed in the study by Scott et al. requires an equal amount of time.

Members of our groups aggregately spend between 0% and 36% of their time in the personal space of other players, with an average of 21%. Number of actions of one person in the personal space of others is reported by [Scott et al., 2004] as being between 0% to 13%. Since that study does not offer the number of total actions per group, we are unable to calculate numbers that are mathematically equivalent to ours.

Our results are inconclusive.

Although our numbers look promising, we feel the difference is not large enough to warrant any conclusion regarding personal space intrusion, considering the fragile basis for comparison. A study in a controlled environment where we set clear tasks and can properly identify users would lead to more conclusive results.

Nevertheless, our overarching goal of high perceptibility of co-location is achieved. Adapting PinguTouch to strongly encourage acting in the personal space of others and generating comparable data could be the focus of another research effort.

PinguTouch achieves high perceptibility of co-location.

6.2 Part two - analyzing enjoyment

6.2.1 Methods

To obtain quantifiable user data, we decided to conduct an opinion survey by using a questionnaire. To maximize visitor participation, it was deliberately designed to fit on one page. This limited us to asking demographic information (age and gender) and five additional questions (Q1 to Q5). Since a considerable part of the museum audience are children and young teenagers, special care had to be taken to make all questions easily understandable.

Entertainment value measured by questionnaire.

- Q1 was aiming to gather a general opinion of the game.
- Q2 asked whether users liked competitive games.
- Q3 asked whether users like cooperative games.
- Q4 asked whether users found it uncomfortable to work in areas occupied by others.
- Q5 asked whether users thought that helping other players handle their pingus was an important part of the game. The question was included to verify our theory that cooperation between players is an important part of the appeal of PinguTouch.

The questionnaire was provided in both German and Dutch, and all participants spoke either one of those as their native language. No additional incentives were provided.

Since previous user observation had already indicated that PinguTouch promotes cooperation, we expected positive

We expected correlations between Q1 and Q3, Q1 and Q5, and Q4 and Q5.

correlation between Q1 and Q3, and Q1 and Q5. The correlation between Q4 and Q5 should be negative, since the questions are opposing each other.

To plan the test, we performed a power analysis as suggested by Cohen [1992]. Based on preliminary trials and the user interviews described in 4.1—"User studies", we decided that the population effect size of the expected correlations should be medium, since users enjoyed themselves and cooperation was clearly visible. We feel that this should reflect in the gathered data.

Power analysis was performed to calculate needed participants.

Hence, using Table 2 from [Cohen, 1992], for a type I error rate of 0.10, a power of 0.80, and expected correlation of $r = 0.30$, at least $N = 34$ participants were needed to get significant results for a one-tail test.

6.2.2 Results

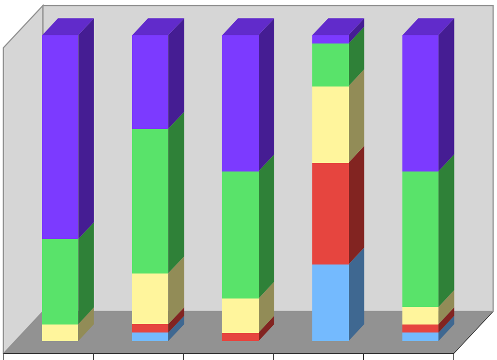
$N = 36$ participants were willing to fill out our questionnaire (39% female). The results are visualized in Figure 6.3. The correlations between the answers to the different questions are shown in Table 6.2.

6.2.3 Discussion

There is an inherent bias in questionnaires like these. Users who enjoyed the game are more likely to spend more time with it and are more willing to assist us in our research. Furthermore, the personal encounter between visitor and developer probably leads to a more positive result than using an anonymous method like mail or internet.

The game was rated highly positive.

For this reason, care needs to be taken when investigating the results. Nevertheless, we feel that the predominantly highly positive feedback to Q1 confirms our general design ideas. The lack of significant correlation between demographic information and Q1 suggests that we accomplished our design goal to create a casual game, i.e., a game that is attractive to a general audience.



	Q1	Q2	Q3	Q4	Q5
🟪 Strongly Agree	24	11	16	1	16
🟩 Agree	10	17	15	5	16
🟨 Neither agree nor disagree	2	6	4	9	2
🟥 Disagree	0	1	1	12	1
🟦 Strongly Disagree	0	1	0	9	1

Figure 6.3: Questionnaire results.

Q5 has led to similarly positive results. The game idea of pingus overcrowding the area of any one single player seems to be well accepted by the players.

The cooperative aspect was deemed very important.

Answers to Q2 and Q3 might suggest that users generally prefer cooperative gaming to competitive gaming. However, we should be careful when making such comparisons. Novelty probably plays a big role here, since only few fully cooperative titles exist (cf. Section 1.1—"Cooperative gaming"). For future studies, it might be interesting to investigate how the results change when asking these questions before and after playing PinguTouch.

Users might prefer cooperative to competitive gaming.

Out of all questions, results of Q4 have the widest distribution. Although only 17% of the participants agreed or strongly agreed that they found it uncomfortable to work in a space occupied by others, the irregularity of answers confirms the results of the research described in Section 2.1.1—"Territoriality in collaborative tabletop workspaces": terri-

Answers to Q4 have the widest distribution.

	Gender	Age	Q1	Q2	Q3	Q4	Q5
Gender	1						
Age	0.21	1					
Q1	0.08	-0.06	1				
Q2	0.12	0.01	-0.15	1			
Q3	0.13	-0.19	0.34	0.36	1		
Q4	-0.28	-0.14	-0.17	0.03	0.08	1	
Q5	0.27	-0.20	0.18	0.03	-0.06	-0.44	1

Table 6.2: Questionnaire correlations.

toriality is deeply ingrained and hard to overcome.

Q1 has relevant correlation to Q3 after outlier removal.

The correlation between Q1 and Q3 was first calculated as 0.24. By removal of one single data row, the value increased to a relevant 0.34. We feel this particular data row to be an outlier (see Figure 6.4). In all other calculations, we kept the data included.

Q1 and Q5 do not correlate relevantly, maybe due to lack of variance.

We could not confirm our theory that Q1 and Q5 correlate relevantly. This comes unexpectedly to us, as helping other players handling their pingus is the primary aspect of the game, and the game itself is very well received. A scatterplot of the data generally confirms that the large majority of participants rated Q1 and Q5 equally (see Figure 6.5). We suspect that the lack of variance in the results of both Q1 and Q5 leads to lack of mathematical correlation in this case.

The correlation of -0.44 between Q4 and Q5 comes as expected, as helping another player handling his pingus necessitates acting in the space used by that player.

The correlation between Q2 and Q3 need further investigation.

A correlation of 0.41 is visible in our data between Q2 and Q3. Since either a positive or negative correlation seems plausible in the real population, we cannot assume a one-tailed test. Further studying with a larger amount of participants would be necessary to get results with a similar significance as the remaining results.

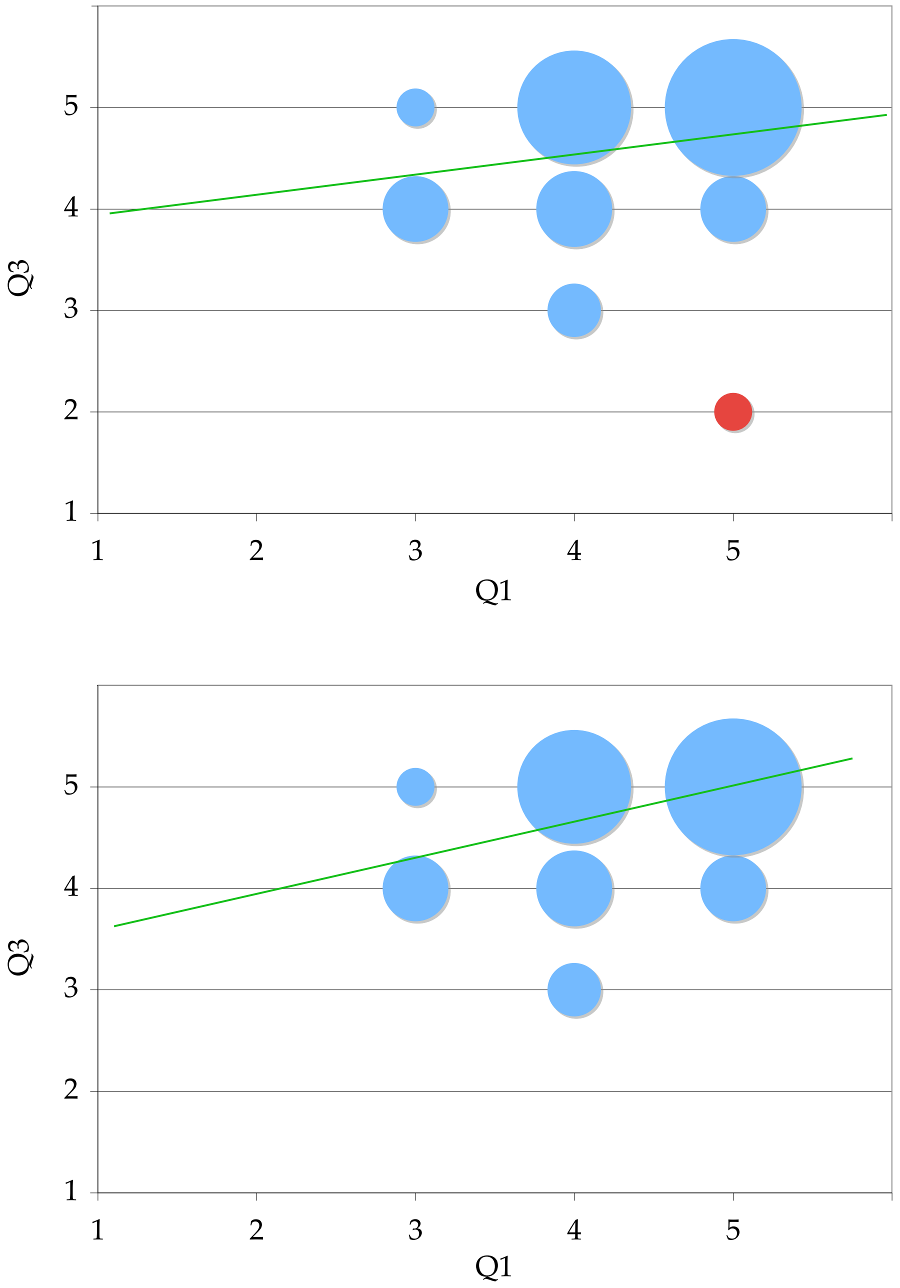


Figure 6.4: Correlation of Q1 and Q3 and regression line. Size of the circles represent occurrences of correlation. **Top:** data including outlier (marked red). **Bottom:** data excluding outlier.

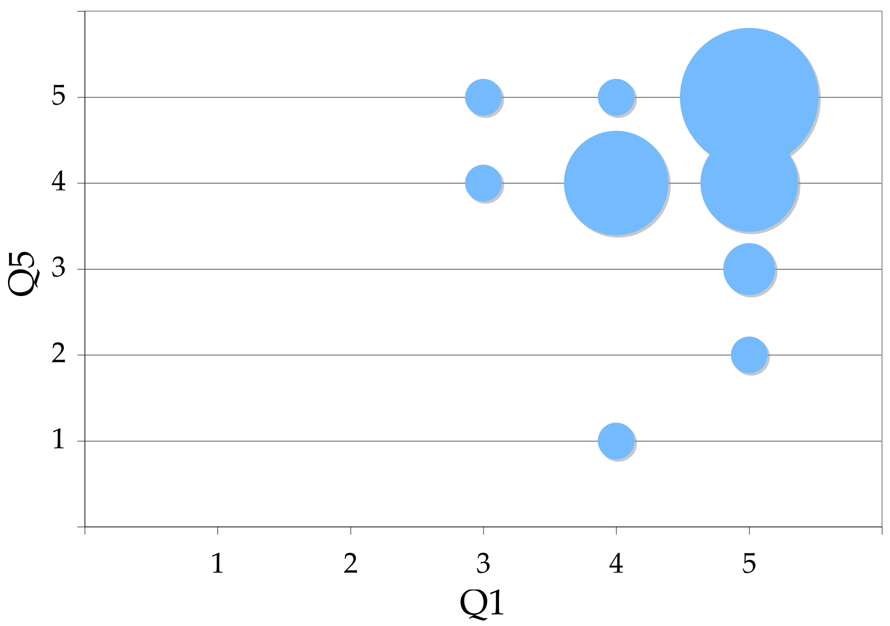


Figure 6.5: Correlation of Q1 and Q5. Size of the circles represent occurrences of correlation.

Chapter 7

Summary and future work

> *"At least I've learnt something from all of this."*
> *"What's that?"*
> *"Never pay more than 20 bucks for a game."*
>
> *—Guybrush Threepwood to Elaine Marley,*
> *The Secret of Monkey Island*

This chapter summarizes the work performed as part of this thesis and suggests several ideas for further development of PinguTouch.

7.1 Summary and contributions

PinguTouch is a casual collaborative tabletop game using multi-touch technology. It is designed to use the unique properties of the employed technology to combine the social experience of a classical board game with the interactive possibilities of modern computer hardware.

The game allows multiple players to lead small penguins, the pingus, across a 2D landscape. Users can perform hand gestures on the table to guide the pingus across obstacles.

The game features a highly real-time environment and encourages close spatial interaction between the players, thus takes advantage of the interactive co-located scenario provided by the tabletop setup.

After describing previous cooperative tabletop applications, we presented our own design ideas and the revisions made after performing user observations. Finally, we conducted user evaluations in form of video analysis and questionnaires to examine whether we reached our design goals. Results show that we were successful in creating an enjoyable game that encourages cooperation.

7.2 Future work

Although PinguTouch is already a complete game in the sense that players gain an enjoyable experience from it, several areas of the game could be enhanced for exploration of further tabletop interactions:

- add cooperative gestures
- add cooperative navigation for increased level size
- include random level events

7.2.1 Cooperative gestures

Cooperative gestures might work better in entertainment context.

To enhance the diversity of the game, more gestures are necessary. In this context, the concept of "Cooperative Gestures" as described in Section 2.3.1—"CollabDraw" could be revisited. Although inappropriate in a task-related environment, allowing even tighter cooperation between users within a single gesture might prove entertaining and would further pronounce the cooperative aspect.

One possible gesture that was thought of, but not implemented, was the "popped balloon"-gesture: one user would have to stretch out a pingu using two fingers of each hand, while another touches the center of the pingu. The pingu would then explode, similar to a needle hitting a balloon. The force of the explosion could remove walls or other obstacles, similar to the original Lemmings game.

7.2.2 Navigation

Change concept of circle to that of spiral offers research opportunity for navigation.

To enable larger levels, yet still allow equal accessibility among all players, the concept of a circular level could be extended towards a spiral. Players could move the view along the spiral by moving the whole hand across the surface, in either the left or right direction.

Parts of a level that are further inside of the spiral could be visible in the background of the active plane, and parts of a level that are further outside of the spiral could be visible as a translucent foreground.

Awareness of level location visible in the current game view and different ways of synchronization between players could offer interesting research opportunities.

7.2.3 Random level events

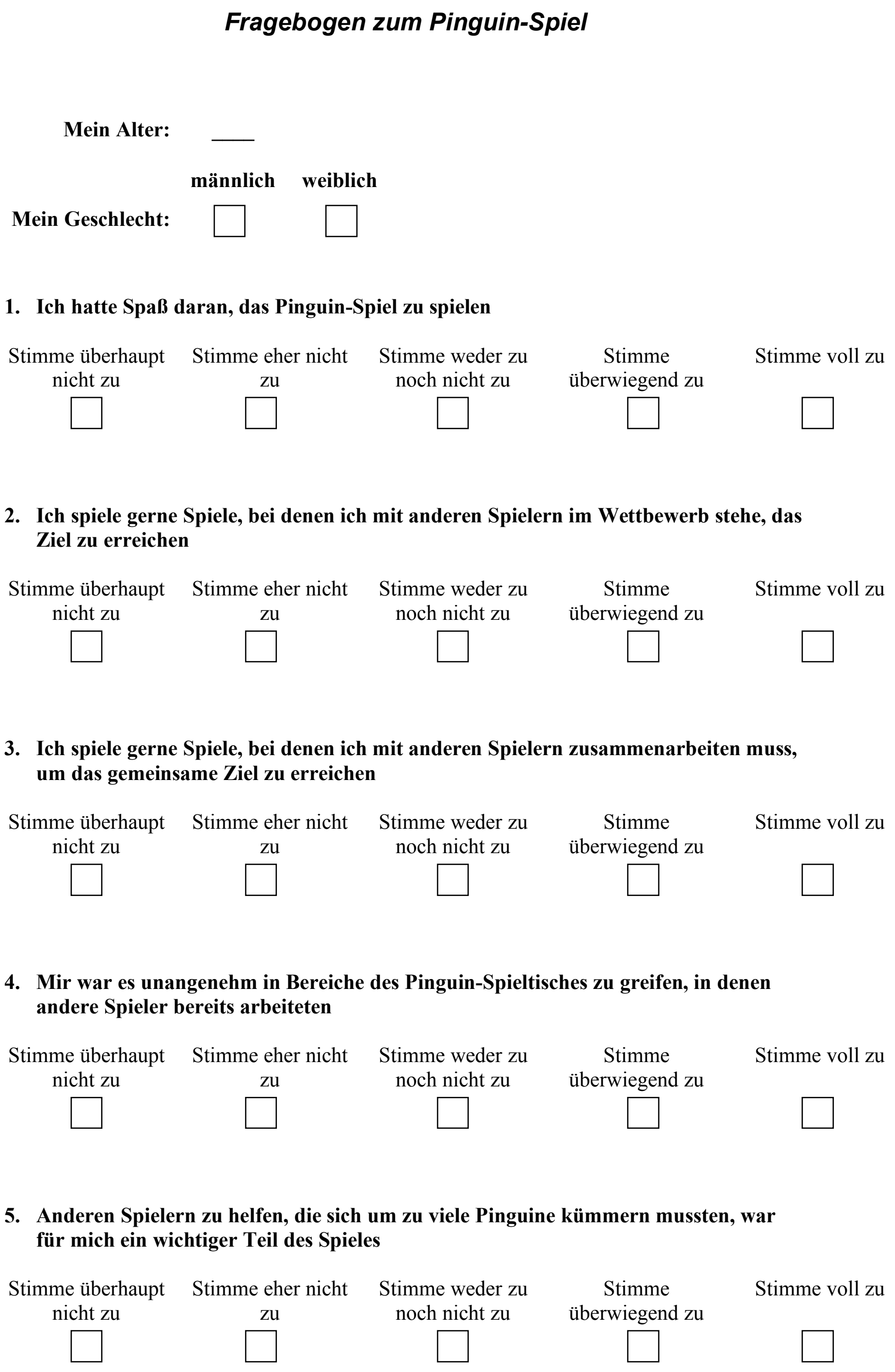
Random level events could be used to research warning signals.

The game experience could be diversified by adding random events involving the game world. For example, rock falls that would crush the pingus could regularly occur at random parts of the level. Players would have to protect the pingus by performing a blocking gestures above their heads.

Since those events would be unpredictable and might not immediately be noticed by the players on the large tabletop screen, they could be used for research into warning signals that alert busy users to a specific location.

Appendix A

Questionnaire



Fragebogen zum Pinguin-Spiel

Mein Alter: ____

	männlich	**weiblich**
Mein Geschlecht:	□	□

1. Ich hatte Spaß daran, das Pinguin-Spiel zu spielen

Stimme überhaupt nicht zu	Stimme eher nicht zu	Stimme weder zu noch nicht zu	Stimme überwiegend zu	Stimme voll zu
□	□	□	□	□

2. Ich spiele gerne Spiele, bei denen ich mit anderen Spielern im Wettbewerb stehe, das Ziel zu erreichen

Stimme überhaupt nicht zu	Stimme eher nicht zu	Stimme weder zu noch nicht zu	Stimme überwiegend zu	Stimme voll zu
□	□	□	□	□

3. Ich spiele gerne Spiele, bei denen ich mit anderen Spielern zusammenarbeiten muss, um das gemeinsame Ziel zu erreichen

Stimme überhaupt nicht zu	Stimme eher nicht zu	Stimme weder zu noch nicht zu	Stimme überwiegend zu	Stimme voll zu
□	□	□	□	□

4. Mir war es unangenehm in Bereiche des Pinguin-Spieltisches zu greifen, in denen andere Spieler bereits arbeiteten

Stimme überhaupt nicht zu	Stimme eher nicht zu	Stimme weder zu noch nicht zu	Stimme überwiegend zu	Stimme voll zu
□	□	□	□	□

5. Anderen Spielern zu helfen, die sich um zu viele Pinguine kümmern mussten, war für mich ein wichtiger Teil des Spieles

Stimme überhaupt nicht zu	Stimme eher nicht zu	Stimme weder zu noch nicht zu	Stimme überwiegend zu	Stimme voll zu
□	□	□	□	□

Figure A.1: Questionnaire used for user evaluation; German version.

Appendix B

Interface definitions

```
class IGestureEffector {
public:
   virtual bool performEffect() = 0;
   virtual int getGestureID() = 0;
   virtual void getGesturePos(int &x, int &y) = 0;
};

class IGestureMatcher {
public:
   virtual std::list<StrokeGroup*> checkForMatches(
      Graph* touchGraph) = 0;
   virtual IGestureEffector* generateEffector(
      std::vector<Vertex>* aGroupMembers) = 0;
   virtual ~IGestureMatcher() {};
};

class WorldObj {
...
public:
   virtual bool fingerDown(int x, int y);
   virtual bool fingerUpdate(Stroke* stroke);
   virtual void fingerUp();
};
```

Bibliography

Christopher Alexander, Sara Ishikawa, and Murray Silverstein. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, USA, 1977.

Tina Blaine and Clifton Forlines. Jam-o-world: evolution of the jam-o-drum multi-player musical controller into the jam-o-whirl gaming interface. In *NIME '02: Proceedings of the 2002 conference on New interfaces for musical expression*, pages 1–6, Singapore, Singapore, 2002. National University of Singapore. ISBN 1-87465365-8.

Jan Borchers. *A Pattern Approach to Interaction Design*. Wiley & Sons, 2001.

Stuart K. Card, Allen Newell, and Thomas P. Moran. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA, 1983. ISBN 0898592437.

Jacob Cohen. A power primer. *Psychological Bulletin*, (112): 155–159, 1992.

Chris Crawford. *The Art of Computer Game Design*. McGraw-Hill/Osborne Media, 1984. URL `http://www.vancouver.wsu.edu/fac/peabody/game-book/Coverpage.html`.

P. Dietz and D. Leigh. Diamondtouch: A multi-user touch technology. In *Proceedings of UIST*, pages 219–226, 2001.

William W. Gaver, Randall B. Smith, and Tim O'Shea. Effective sounds in complex systems: the arkola simulation. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 85–90, New York, NY, USA, 1991. ACM Press. ISBN 0-89791-383-3. doi: http://doi.acm.org/10.1145/108844.108857.

Edward T Hall. *The Hidden Dimension*. Anchor Books, 1966.

Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-271-2. doi: http://doi.acm.org/10.1145/1095034.1095054.

D. Johnson and R. Johnson. Positive interdependence: Key to effective cooperation. In R. Hertz-Lazarowitz and N. Miller, editors, *Interaction in cooperative groups: the theoretical anatomy of group learning*. Cambridge University Press, Cambridge, 1992.

Michael Kipp. Anvil - a generic annotation tool for multimodal dialogue. In *Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech)*, pages 1367–1370, 2001.

Carsten Magerkurth, Maral Memisoglu, Timo Engelke, and Norbert Streitz. Towards the next generation of tabletop gaming experiences. In *GI '04: Proceedings of the 2004 conference on Graphics interface*, pages 73–80, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society. ISBN 1-56881-227-2.

Carsten Magerkurth, Adrian David Cheok, Regan L. Mandryk, and Trond Nilsen. Pervasive games: bringing computer entertainment back to the real world. *Comput. Entertain.*, 3(3):4–4, 2005. ISSN 1544-3574. doi: http://doi.acm.org/10.1145/1077246.1077257.

Nobuyuki Matsushita and Jun Rekimoto. Holowall: designing a finger, hand, body, and object sensitive wall. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 209–210, New York, NY, USA, 1997. ACM. ISBN 0-89791-881-9. doi: http://doi.acm.org/10.1145/263407.263549.

Meredith Ringel Morris, Anqi Huang, Andreas Paepcke, and Terry Winograd. Cooperative gestures: multi-user gestural interactions for co-located groupware. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1201–1210, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-372-7. doi: http://doi.acm.org/10.1145/1124772.1124952.

Anne Marie Piper, Eileen O'Brien, Meredith Ringel Morris, and Terry Winograd. Sides: a cooperative tabletop computer game for social skills development. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 1–10, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-249-6. doi: http://doi.acm.org/10.1145/1180875.1180877.

Tran Cong Thien Qui, Ta Huynh Duy Nguyen, Asitha Mallawaarachchi, Ke Xu, Wei Liu, Shang Ping Lee, Zhi Ying Zhou, Sze Lee Teo, Hui Siang Teo, Le Nam Thang, Yu Li, Adrian David Cheok, and Hirokazu Kato. Magic land: live 3d human capture mixed reality interactive system. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1142–1143, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-002-7. doi: http://doi.acm.org/10.1145/1056808.1056853.

Jef Raskin. *The Humane Interface. New Directions for Designing Interactive Systems.* Addison-Wesley Longman, Amsterdam, March 2000.

Derek F. Reilly and Kori M. Inkpen. White rooms and morphing don't mix: setting and the evaluation of visualization techniques. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 111–120, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-593-9. doi: http://doi.acm.org/10.1145/1240624.1240640.

Stacey D. Scott, M. Sheelagh, T. Carpendale, and Kori M. Inkpen. Territoriality in collaborative tabletop workspaces. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 294–303, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-810-5. doi: http://doi.acm.org/10.1145/1031607.1031655.

Anthony Tang, Melanie Tory, Barry Po, Petra Neumann, and Sheelagh Carpendale. Collaborative coupling over tabletop displays. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1181–1190, New York, NY, USA, 2006. ACM Press.

ISBN 1-59593-372-7. doi: http://doi.acm.org/10.1145/1124772.1124950.

E. Tse, S. Greenberg, C. Shen, and C. Forlines. Multimodal multiplayer tabletop gaming. In *PerGames 2006*, May 2006.

Mike Wu and Ravin Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 193–202, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-636-6. doi: http://doi.acm.org/10.1145/964696.964718.

Index

Typeset December 17, 2007