BACHELORARBEIT

Adaptation and Evaluation of Numpty Physics for Multi-Touch Multi-Player Interaction

Zur Erlangung des akademischen Grades
Bachelor of Science (BSc.)

ausgeführt am
Institut für Rechnergestützte Automation
Forschungsgruppe Industrial Software

der Technischen Universität Wien

unter der Anleitung von
Univ.-Prof. Dipl.-Ing. Dr. Thomas Grechenig

durch:
Thomas Perl
e0725603@student.tuwien.ac.at
und
Stefan Kögl
e0725134@student.tuwien.ac.at

Wien, August 2009

Abstract

The aim of this project was to enhance the overall user experience of the popular game Numpty Physics (Edmonds, 2008) through the use of multi-touch input.

The user interaction consists of drawing, moving and deleting shapes with a stylus on touch-screen devices or a mouse on desktop computers.

In order to allow multi-player abilities and multi-touch gestures in the game, we modified Numpty Physics to work with multi-touch finger input.

New features leveraging the available software and hardware infrastructure, such as different multiplayer modes and new levels were also implemented.

We added sound output to the game in order to increase the amount of feedback that the user gets, which improved the usability of the game.

Evaluations of incremental subsets of the functionality have been carried out in various stages during development. The results have been fed back into our development process allowing us to quickly react to discovered problems.

Our multi-touch version of Numpty Physics has been well received by testers. User acceptance of the multi-touch version was higher than that of the regular mouse-driven version, mostly because of the more natural interaction with the game world, and the possibility for two persons to play simultaneously.

Keywords

Multi-Touch, Numpty Physics, Gesture Recognition, Multi-Player, Physics Game

Contents

List of Figures

List of Tables

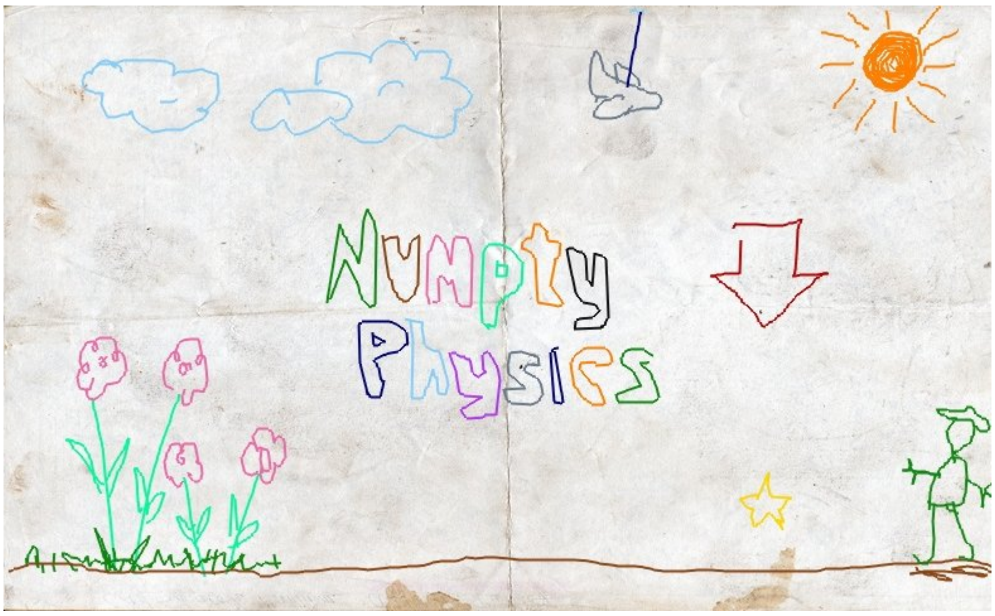

Figure 1: The title screen of Numpty Physics

1 Introduction

With multi-touch technology becoming more prominent lately, there has been much discussion about its suitability for certain use cases. To highlight real-world porting issues of existing software to multi-touch technology, we decided to equip an existing single-user, single-touch game with multi-touch control and use the potential of this technology for further enhancements.

We chose to use the well-known puzzle game Numpty Physics, because we thought it could benefit a lot from multi-touch technology. Its availability as open source made the choice even easier.

1.1 Existing Software

Numpty Physics (Edmonds, 2008) has been created in March 2008 by Tim Edmonds as a physics-based drawing game for Maemo-based Internet Tablets with single-touch touchscreen input. These Internet Tablets have a 4.1 inch screen with a resolution of 800x480. As a result of the small screen, the game is not playable with fingers, but only with the included stylus and uses modifier keys to access certain functions. The game was well received by the Tablet community for its ease of use and challenging,

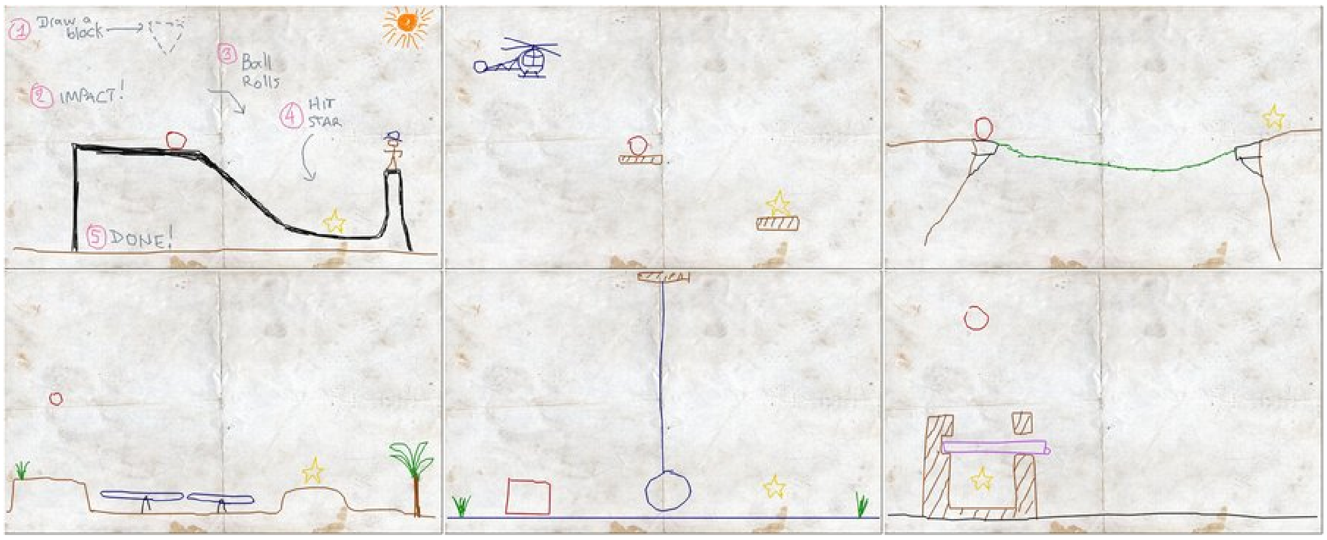

Figure 2: The first six single-player levels of the original game

but enjoyable levels that provide quick satisfaction due to their flat learning curve. The title screen of Numpty Physics (which is also playable as a level) is shown in figure 1.

The gameplay of Numpty Physics is described on its homepage as follows:

> Harness gravity with your crayon and set about creating blocks, ramps, levers, pulleys and whatever else you fancy to get the little red thing to the little yellow thing.

The full-screen game area does not restrict what you can draw, but you can only move objects that you have drawn yourself. Moving of other objects is achieved by drawing triangles that fall down and hit the objects or by creating swings, spoons, hammers and other objects. There is no single right way of solving a level, but an unlimited number of possibilities, just as there is an unlimited number of possibilities to move one object to another place in the real world.

The game consists of several independent levels (the first levels are depicted in figure 2). In each level, the user's task is to get the red "token" circle, to the yellow "goal" star (figure 3). The challenge here is that both objects can't be interacted with directly. Instead, the user tries to accomplish his task by drawing new objects which, once created, follow basic laws of physic, such as gravitation. Drawn objects fall to the ground, hit other objects and set them in motion. Based on this simple interaction, the user is required to fulfill complex movements with his token to reach the goal.

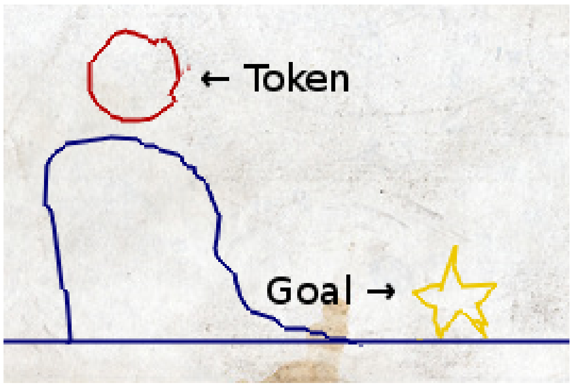

Figure 3: The token (in red) and the goal (in yellow) in Numpty Physics

New levels can easily be created by switching the running game into an “edit mode” which allows users to design new, challenging levels that can then be sent to the author and published on the game's website[1]. In edit mode, the user can draw several different types of objects with different behavior:

- normal - This is the same behavior as strokes drawn during the normal game mode
- sleeping - The object does not move until it is hit by another object
- fixed - The object never moves and is fixed in space (ground, etc...)
- decoration - The object does not have physical properties in the game world (text, background art, etc...)

The existing software provided support for only one simultaneous drawing operation due to the single-touch characteristics of the target devices. Deleting and moving user-drawn strokes was only possible by using the stylus in combination with hardware keys on the Maemo-based devices, which is not intuitive and proves to be hard with one hand holding the device and the other one operating the stylus.

In addition to the original version, simple ports of the game to other platforms exist, including Linux, Windows, Mac OS X, FreeBSD and Solaris. All these ports use the computer mouse as the pointing and drawing device, what can be tedious for the tasks at hand, and for drawing in general.

[1]http://numptyphysics.garage.maemo.org/levels

The mouse is further used for deletion and movement of strokes with the middle and right mouse buttons. Again, simultaneous actions are not allowed due to the single mouse input mechanism on these platforms. This effectively makes the original Numpty Physics a single-player, single-touch game.

The physics engine used in the game is Box2D (Catto, 2007), which is also used in many other physics-based games (see related work).

Numpty Physics is licensed under the GNU General Public License Version 3, allowing us to modify the source code and publish the results.

1.2 Goals and Concept

The high-level goal of this project was to improve the user-experience of Numpty Physics by leveraging multi-touch technology. An evaluation of the current software highlighted difficulties in the gameplay for which we developed enhancements using multi-touch functionality.

Our idea was to first transform the game's basic operations – drawing, moving and deleting – to intuitive multi-touch input and carry out our first evaluation of the modified game. We decided to test our modifications very early in the development process to quickly detect usability issues.

Taking into account the results of the first evaluation, we decided to create multi-player modes and add new features using multi-touch input.

The resulting implementation was then tested in the second evaluation to ensure high usability of the new game. We also used the second evaluation as an opportunity to uncover remaining problems with the multi-touch interaction.

1.3 Evaluation

With the focus being on "user experience", measuring success is a very subjective matter. Hence evaluations were based on both user interviews and our own observations.

Multi-touch features that were intended as improvements of known issues of the original game were evaluated in direct comparison to the existing software. Subjects were asked to play Numpty Physics either on a PC or handheld device and then moved on to the multi-touch version.

Our measure on the effectiveness of newly introduced multi-touch features in the game were therefore based on the acceptance of the features by the test subjects. "Thinking Aloud" was used during the evaluation to find problems and assumptions that the test subjects made while playing the game. For tests with multiple players, we used the "Co-Discovery" method in addition to "Thinking Aloud".

2 Related Work

Several papers have already been published in various fields of research related to our work. These range from the foundations of multi-touch, such as devices and concepts, to collaborative multi-touch games. As our projects implicitly build upon many of these works, we will describe relevant work and contrast our project with existing software and highlight special fields of interest.

2.1 Multi-Touch Technology

Although researched for a relatively long time, multi-touch has gained much popularity recently. Interest in this topic has been fueled by the integration of multi-touch technology into mainstream products and research continues to deliver new ideas and concepts.

The history of multi-touch technology dates back to 1982 when the first multi-touch device was introduced by Mehta (1982). It used a frosted-glass panel on which black spots generated by finger pressure were tracked and was used for multi-touch picture drawing.

In the following year a paper (Nakatani and Rohrlich, 1983) discussing properties of touch-based input devices was published. Although not directly related to multi-touch it deals with “soft machines” which ought to be operated without “hard” controls like keyboards and hardware buttons but rather with “soft” interface elements that are displayed on an interactive surface which adapts to the current state of the program flow. The intention was to simulate purpose-built machines which controls match their designated functions with general-purpose computers.

Following these initial works, research for multi-touch has developed in various areas including image processing, mobile devices and productivity software. Buxton (2007) provides a thorough overview of early multi-touch projects.

With these developments several technologies emerged and are used today, each with specific advantages and disadvantages. Selecting a suitable technology depends on various factors, such as the purpose of the device, intended audience and contraints imposed by the environment, for example available lightning. The current technologies for detecting user contact can be roughly divided into two categories.

Capacitive methods signal tracked points by detecting touches of condutive items, such as fingertips. Devices using capacitive technologies require grids of sensing lines or electrodes below the surface that are used to determine the points of contact. This makes the assembly of such devices

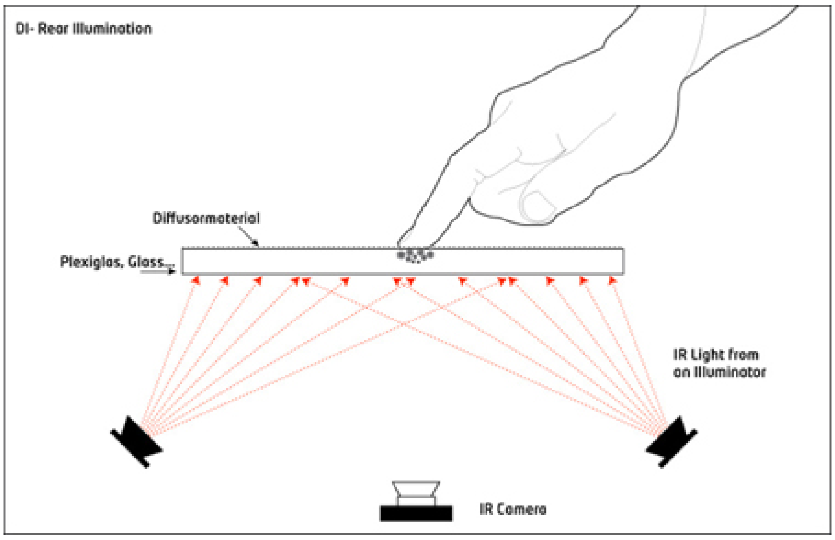

Figure 4: Schematic Diagram of Rear Diffused Illumination from NUI Group (2009a)

very complex and expensive but allows very exact positioning. Hence they are currently mainly used for commercial products in combination with LC displays.

The second category uses **optical recognition** of shadows produced by fingers on the devices' surfaces. Capturing is commonly done by a video camera positioned below the surface. To avoid interference with other light sources such as a projector for visual output, infrared light is captured by the camera. The recorded images are then forwarded to a tracker software which uses means of image processing to separate touching points from the background. Optical Methods are popular for research purposes as devices using such methods can be assembled relatively easy and are quickly adapted to changing needs. Drawbacks such as the space needed for the projector and the video camera often play a minor role in research setups.

A very common optical method, depicted in figure 4, that was also utilized for our evaluations is Rear Diffused Illumination. In this approach a diffusing material, such as plexiglass, is used as the touching surface. Infrared light sources are placed below and directed towards the panel. When the surface is touched the infrared light is reflected and captured by a camera which is also positioned below the panel.

Another optical method with widespread use is Frustrated Total Internal Reflection (Han, 2005), which injects infrared light into a plexiglass pane that is also used as the projection surface. Total

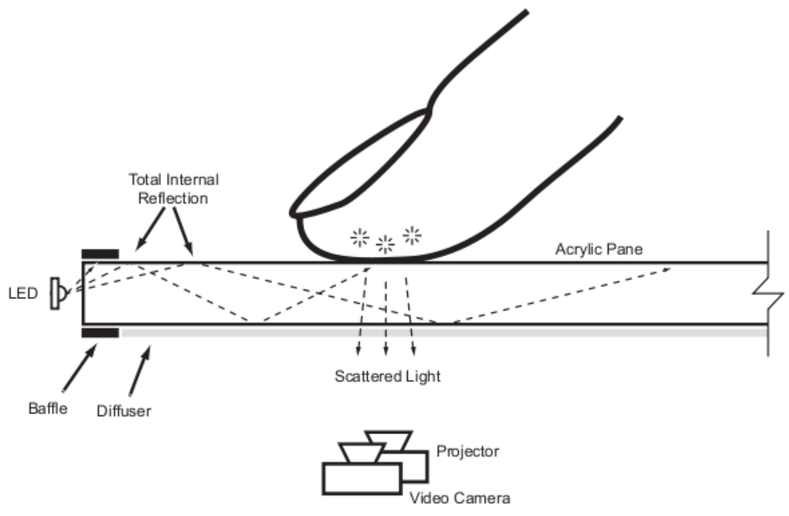

Figure 5: Schematic Diagram of Frustrated Total Internal Reflection from Han (2005)

Internal Reflection keeps the light beams inside the glass. At the point where a user's finger touches the surface, the light is reflected and scattered outside the pane where it is captured by the video camera.

Another optical method is Laser Light Pane Illumination with laser beams arranged in a grid directly on the surface of the pane. When a finger touches a spot on the surface it disrupts the laser beams crossing the spot. From the horizontal and vertical coordinates of the disrupted lasers the touching points can be calculated. A major disadvantage of this technology is that it does not support two separate touches in the line of a laser. This can be reduced be placing redundant lasers in the corners, but can not be prevented completely.

2.2 Collaborative Games

Khaled et al (2009) gives an overview of various aspects of collaborative games in multi-touch environments. The researchers developed two games specifically for evaluating collaborative gameplay on multi-touch tables. One of these games – "The Laundry Game" – is shown in figure 6.

According to Khaled et al (2009), there are two main reasons why multi-player games are suited for multi-touch tables:

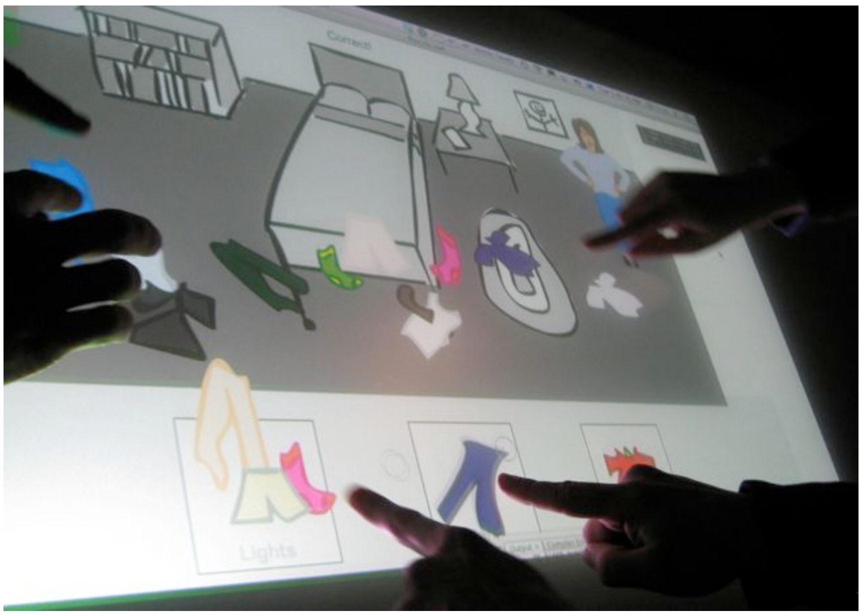

Figure 6: “The Laundry Game” by Khaled et al (2009)

> Firstly, the system makes it easy because the surface allows several people to all physically participate and interact simultaneously. Secondly, ordinary tabletops are a culturally and socially established support artifact and environment for many collaborative activities from food preparation to games to meetings.

The task of The Laundry Game (Khaled et al, 2009, page 3) is to sort clothing items into different bins based on the color of each item. Scores are only kept for a single player, but multiple players can cooperate to gain a better score, but playing against each other is not possible. In order to enhance the fun factor of our game, we added both cooperative and versus game modes to Numpty Physics.

The Laundry Game makes use of sound clips to provide feedback to its users about successful move actions and the end of the game. In order to increase the usability and feedback of Numpty Physics, we decided to add sound, too.

“Labour of Loaf” is the name of the second game that has been created and evaluated by Khaled et al (2009). The goal of this game is to construct sandwiches corresponding to on-screen orders. Scoring can be toggled between a group mode in which all players collaborate for a good score and an individual mode where each plate is scored separately. This allows for both cooperative and versus

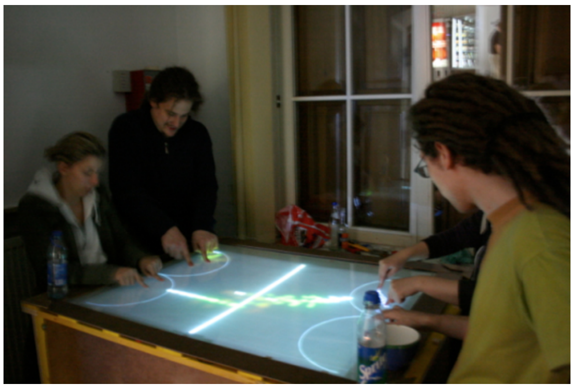

Figure 7: Four users playing the Puh game on the cueTable (Gross et al, 2008)

game modes.

In contrast to the purpose-built games in Khaled et al (2009), our studies have been based on a pre-existing game, which has been extended with multi-touch input capabilities during the project. We believe that due to the vast amount of existing games, adding multi-touch input to existing games will become more important as multi-touch hardware matures and becomes more affordable.

Khaled et al (2009) used tbeta (NUI Group, 2009b) for tracking points (which is the same software we used). The games were written in Flash, whereas we used C++ for the game core and Python for multi-touch processing.

Our findings partially overlapped with Khaled et al (2009), especially in the areas of user behavior and technical constraints. As an example, hardware latency was an issue that affected both projects.

In Gross et al (2008) the authors discuss their development of the cueTable, a tabletop for cooperative and competitive interaction. Figure 7 shows four users playing the Puh game, a game similar to Atari's Pong.

The gameplay of the Puh game is described by its authors as follows:

> [The game is] played by two teams each consisting of one or two players. Teammates stand next to each other at one short edge of the table, facing the other team. All players have personal playing zones, where they create paddles with their fingers to shoot the ball towards the other team's goal line.

By utilizing dedicated playing zones for each player, the Puh game allows for identifying players by assigning touches in these zones to pre-defined players.

In order to avoid the complexity and restrictions that come with player identification on a multi-touch table without hardware support for assigning touch points to players, we decided not to require player identification in Numpty Physics.

Latency has been an issue with cueTable, but after a short period of playing the game, the users learned to work around this issue by reacting to the ball movements in advance. We experienced similar issues with latency in our project.

Compared to other multi-touch projects described in this section, Gross et al (2008) developed their own "MultiTouchEngine" instead of re-using existing tracker software such as tbeta (NUI Group, 2009b). According to its authors the events generated by the MultiTouchEngine

> [...] are compatible with the existing Java concepts of AWT and Swing. They are treated like mouse events and can be used in any existing Swing application.

This approach allows Java developers using Swing and/or AWT for their GUI to directly run mouse-based applications on the cueTable. A disadvantage of this approach is that programming languages and environments other than Java are not directly supported. In contrast, TUIO (Kaltenbrunner et al, 2005), which is used by many existing multi-touch projects, supports several different languages and environments.

2.3 Multi-Touch Physics Games

Multitouch Barcelona (2008) created a multi-touch clone of Crayon Physics in May 2008. The resulting application is more of a technology demonstration with physics effects rather than a full game, although users can try to draw objects and create games using these objects (see figure 8, where two developers are playing ping pong using the game).

The game itself features no levels, but only provides the necessary UI and bindings to allow users to create objects and interact with them using a multi-touch table.

Their website[2] describes the application's principle:

> [...] an interactive drawing canvas where people can use a brush or their own fingers to draw shapes that turn into physical objects on the screen.

[2]http://www.multitouch-barcelona.com/?page_id=385

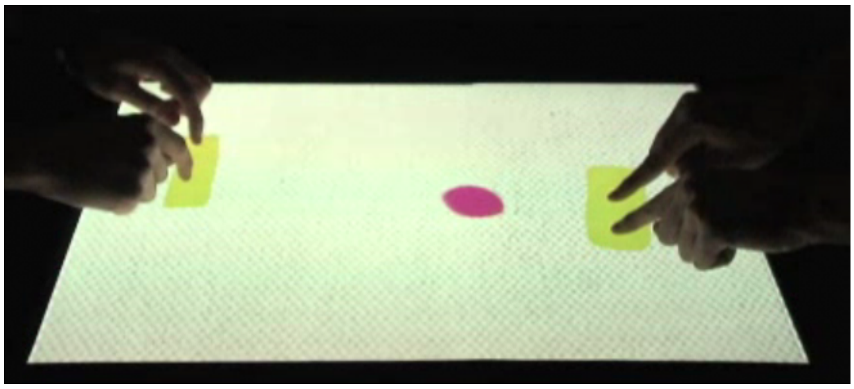

Figure 8: “Multi-touch Crayon Physics” by Multitouch Barcelona (2008)

In contrast to our Multi-touch Numpty Physics game, the hardware used by Multitouch Barcelona (2008) seems to be more accurate, as drawing shapes with a brush on the table is also possible. We have not tried to use brushes on our table, as we added two- and three-finger-gestures to the game, and switching back and forth between the brush and fingers would be tedious.

Multi-touch Crayon Physics has been released in both binary and source versions on their homepage[3]. The application is written in ActionScript 3 and the source code is released under the terms of the Creative Commons Attribution–Share Alike 3.0 Unported license.

Hartman (2008) wrote a multi-touch physics game called “Station Defender” (shown in figure 9) using the NUI Framework at Luleå Tekniska Universitet. The goal of the game is explained as

> [...] to survive for as long as possible, there is a enemy station in the middle that shoots lasers, plasmas and missiles in all directions and the players have to survive by building different kinds of walls that will protect him/her from the shots.

The NUI Framwork used in “Station Defender” is written in C++ and uses OpenGL for its graphical output. In comparison, Numpty Physics is also written in C++, but uses the SDL library for its graphical output, which makes it runnable on devices which are not able to run OpenGL-based software due to lack of 3D hardware or slow speed in case of software rendering.

Just like Numpty Physics, both of these multi-touch physics games (Multitouch Barcelona, 2008; Hartman, 2008) use the Box2D engine (Catto, 2007) for their physics simulation. We suggest that

[3]http://nuigroup.com/forums/viewthread/1919/

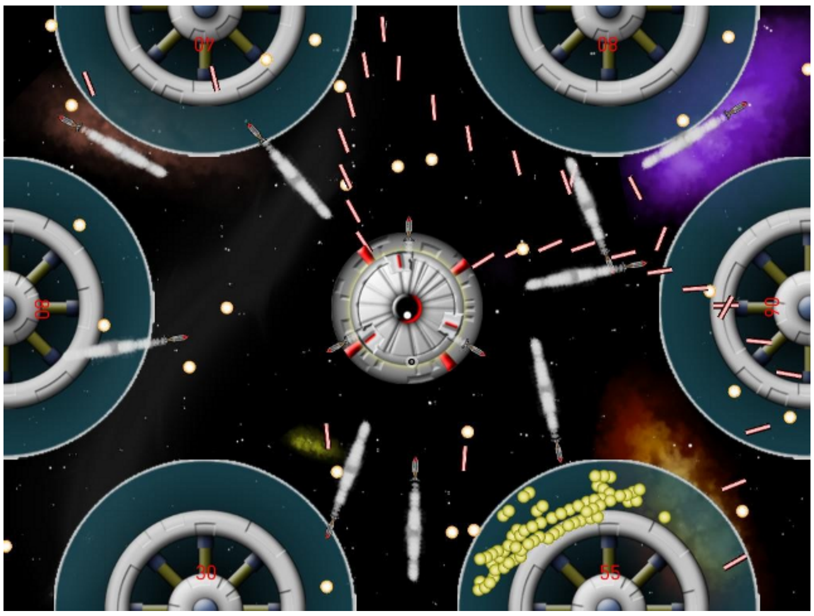

Figure 9: “Station Defender” by Hartman (2008)

for multi-touch games that make use of physics, the Box2D engine is well-proven and can be recommended for future projects. In addition to the original C++ implementation, ports for ActionScript 3, JavaScript, C#, Java, Python and Ruby are available, allowing for a wide range of programming languages to choose from.

2.4 Increasing Usability with Sound

According to Brewster and Crease (1999), adding sound to existing graphical user interfaces can improve usability. Although Brewster and Crease (1999) deal mostly with menu-based interfaces, we decided to add sound to Numpty Physics before the second evaluation. We then determined the gain in usability by comparing the results of the first evaluation (without sound) with the results of the second one and found that users were more confident in knowing that their actions were carried out, even if they could not confirm the actions visually.

As a reason for the improved usability, the authors explain that

> [...] sound combined with graphics can significantly improve usability by taking advantage our natural ability to share tasks across sensory modalities [...]

There are several problems with multi-touch input that decrease the amount of user feedback compared to using, for example, mouse-based interfaces (see figure 14 for an example of this in Numpty Physics). The addition of sound makes it possible to re-add some of the feedback to the UI.

Errors in user input are split into two different categories in Brewster and Crease (1999):

- The wrong menu item was mistakenly selected (the user chose the wrong item)
- The cursor slipped onto the wrong menu item accidentally as the button was being released (an item slip)

In case of Numpty Physics, we deal with touch operations instead of clicks and selections. There is no real “wrong” way of drawing strokes, and drawing a wrong stroke does not have the same fatal consequences as selecting a menu item (e.g. “delete document”). The “item slip” that is described in Brewster and Crease (1999) can be compared to false blobs (touches that are detected by the hardware that the user did not carry out, and touches that are carried out by the user but are not detected by the hardware) in multi-touch applications. The “item slip” in menus is hard to realize for the user, as most menu-driven UIs hide the menu as soon as an item has been selected. In the case of multi-touch interfaces, false blobs are not physically seen or caused by the user – but the results affect the game world, leading to user frustration (“Why did [the game] draw this line?”).

To quote Brewster and Crease (1999), page 5:

> [...] the right feedback must be given to ensure users know what is going on.

For the delete stroke action (three fingers touch the stroke that is to be deleted), we play back an explosion sound for each stroke deleted. Due to the pointing nature of this gesture, the stroke that is to be deleted happens to be covered by the user’s hand in almost all cases. Hearing the explosion sound lets the user know if and when a stroke is deleted - even if it does not give a clue which stroke has been deleted.

Another interesting aspect of using sound in games is annoyance. Adding too much sound (or the wrong sound effects) could cause the player to feel uncomfortable and irritated when playing the game. We chose subtle sound effects for our game, and the persons that played the game during the evaluation were only exposed to the sounds for a duration of 15 minutes. It has to be determined if playing sounds during a longer game session would make the persons playing the game feel annoyed.

Brewster and Crease (1999) cite excessive intensity of sound as the main cause of annoyance. Annoyance due to speech sounds is also mentioned, but is not relevant for our work, as we have not added speech output to Numpty Physics.

We incorporated methods for avoiding annoyance by using good quality loudspeakers and positioning them in the corners of the table (see figure 18), which provides improved localization of the sound source (Brewster and Crease, 1999).

Gärdenfors (2003) analyzed the design of sound-based games for visually impaired users and points out important aspects of developing such games. In sound-based games, sound plays the primary role of output, whereas in our multi-touch game, sound is only used as an additional feedback mechanism. Indeed, multi-touch hardware has its focus on good visual feedback, as this is the way how users interact with the surface.

The reason why we chose to consult Gärdenfors (2003) in our work and refer to it here in this section is because it shows in great detail how the information carried by sound effects can be maximized. Choosing the right sounds is important in this regard:

> An obvious method is to use actual sound recordings of the event one intends to illustrate. However, since all objects or events do not emit sounds, authentic sounds are not always available, and when they are, they are not always recognizable.

In case of our game, we use the sound of a chalk scribbling on a blackboard as the sound for drawing events, which is obvious for the user. In the case of dragging strokes, we had to improvise a little, as dragging objects through the air does not normally make a sound in the real world. The same thing is true for deleting strokes - in the real world it is not possible to have things disappear from one moment to the other. We dealt with this problem by using sounds that seemed to work well for the task at hand. See table 2 for a list of sound effects used in the final version of the game.

The multi-player nature of our game made it necessary to simultaneously give feedback to different players on the same multi-touch table. In order to achieve this separation of feedback on the table, we panned the sound effects according to the X position of the event that caused the sound effect.

Gärdenfors (2003) describes the positional separation of sounds using stereo speakers:

> Sounds can be separated spatially in a standard stereo sound system. However, as stereo only represents one dimension, the images that can be conveyed are limited. If one wants

> communicate spatial structures closer to the complexity of graphics on a two-dimensional computer screen, some kind of surround sound system is needed.

We did not chose to add surround sound to the game, as the sound effects are only needed to enhance the feedback for the user, and the effort it would have required to set up and implement surround sound might not have improved the user experience of our game.

	Stylus-based input	Mouse input	Multi-touch input
Draw a new line	Draw	Left button	Single finger
Move existing line	Hold “zoom in” button and draw	Right button	Two fingers
Delete a line	Hold “zoom out” button and touch	Middle button	Three fingers

Table 1: Actions and their input mappings for all three input methods

3 Implementation

3.1 Porting Existing Functionality

As our goal required that Numpty Physics could be fully played on a multi-touch table without the need to interact with conventional input mechanisms like a mouse, keyboard or Stylus, porting the existing functionality to multi-touch input was essential.

The most important interactions with the game are drawing, moving and deleting objects, but further actions, mostly for controlling the flow of the game are also available. These include pausing and resuming the game, switching levels and opening the menu.

In order to make the transition to multi-touch straightforward and to utilize the existing gameplay rules, we decided to map the mouse buttons of the original game to grouped finger gestures. We used the findings of Matejka et al (2009) as a reference for our own implementation, but used a simpler approach as described in table 1 in order to not depend on the order in which the fingers touch the table.

The most basic action – **drawing shapes** – was assigned the most intuitive gesture: drawing with a single finger. Each shape drawn with one finger directly resulted in the corresponding shape being created within the game world.

Moving shapes was assigned to a dragging gesture with two fingers.

Deleting a shape was assigned the gesture of tapping the shape that is to be deleted with three fingers. This, as it is the most destructive interaction, uses the most number of fingers simultaneously to avoid misinterpretation of the input by the application.

Actions that were previously carried out by using keyboard shortcuts were assigned gestures that activated these features and had to be designed to be hard to confuse with drawing gestures.

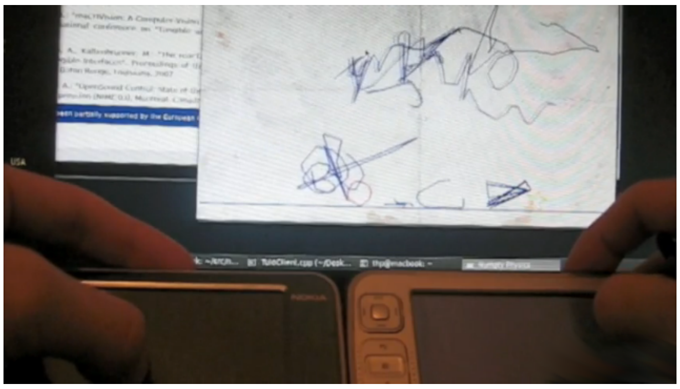

Figure 10: Using two single-touch touchscreen devices to simulate multi-touch

3.2 Multi-Touch Input

For getting the multi-touch input data into our code, we used TUIO (Kaltenbrunner et al, 2005), which is a widely-used protocol for multi-touch input events. Using TUIO, we are compatible with a wide range of tracker applications without having to write a separate input module for each tracker.

For the tracker software, we used Community Core Vision (NUI Group, 2009b) (the application was formerly known as “tbeta”). This application takes care of converting the video input received by the camera to cursors that can be tracked over time with their coordinates using a unique ID for each cursor.

3.2.1 Multi-Touch Input using the MTmini

For testing our input-handling code and trying out gestures at home, we built two MTmini devices (using instructions from Sandler, 2008) with Sony PS3-eye cameras due to their support in Windows, Mac OS X and Linux operating systems, although the tracker software we used was only able to utilize the camera in Mac OS X. We used the same tracker software for both the MTmini and the real multi-touch table - this way, changing the environment in which our software ran just required setting the parameters of the tracker software for the corresponding input device without needing to rewrite the software.

3.2.2 Multi-Touch Input Simulation using Single-Touch Tablets

For developing the software at home and testing our multi-touch input code, we also customized TUIO_CPP's SimpleSimulator application to run on Maemo devices and send TUIO messages over the network. Running our modified version of the SimpleSimulator - which we called TuioTablet to differentiate between the original version - on two tablets simultaneously (see figure 10) allowed us to simulate up to two multi-touch touch points. We did not use this technique for evaluations, but it was helpful in developing the multi-touch software when not having access to the table.

The software TuioTablet that has been created as a byproduct of our development work can be downloaded from our multi-touch website (Perl and Kögl, 2009).

3.3 Software Architecture

In order to facilitate quick experimentation and a more object-oriented approach to developing the gesture recognition and cursor grouping, we chose to use the Python programming language to develop our multi-touch input module. As Numpty Physics itself is written in C++, we embedded the CPython interpreter in the main application and wrote some glue code to receive events from the Python code and put them on the SDL event queue where they were processed in the main loop of the game, as can be seen in figure 11.

The routines for handling the mouse events were extended to accept a "cursor id" as additional parameter. The standard mouse input always passes the value 0 to these functions, while the multi-touch input module passes values starting from 1 up to the maximum number of allowed tracking points. This ensures that both mouse and multi-touch input can be used simultaneously and are handled through the same code.

In order to receive and parse the TUIO messages, we used the open source PyTUIO (Leidel, 2007) library in our input module. Unfortunately, the PyTUIO library just provides code for parsing the messages and does not include support for fixing tracking errors such as false blobs. Gesture recognition and cursor grouping are also not supported by PyTUIO at the moment, so we wrote the required features on top of PyTUIO in our multitouch tracking code.

To keep the architecture simple and make it compatible with the widest range of multi-touch trackers possible, we use a simple one-way event architecture for passing events between the game and the multi-touch code. This means that the multi-touch module's task is to interpret its input without knowing about the current state of the game.

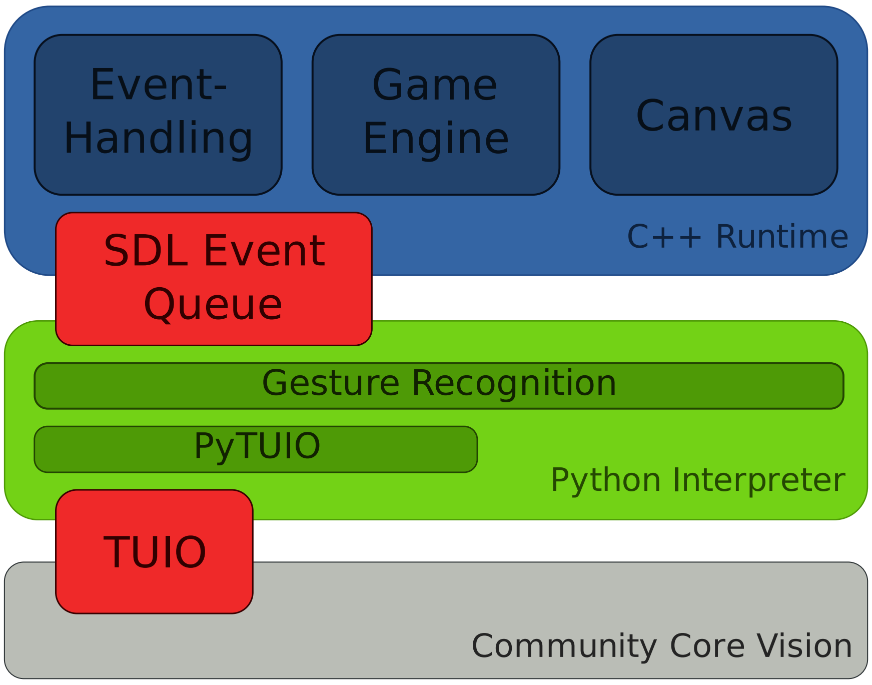

Figure 11: Multi-touch Numpty Physics Architecture Overview

3.4 Gesture Tracking

For integrating the multi-touch events with the mouse events, the concept of cursors was introduced, of which multiple can be active at the same time. The existing mouse handling code was adapted to serve as a generic event handler, which enabled simultaneous processing of multi-touch and mouse input events.

The solutions in Matejka et al (2009) describe methods for implementing button chording (e.g. allowing the left and middle mouse button to be pressed simultaneously), which is not needed for Numpty Physics. Instead, we opted for a simple cursor-grouping method which is easier to implement and faster to detect the amount of fingers and the start of an interaction.

The input module has to delay sending events to the game until it is known if the user wants to draw, move or delete a stroke (i.e. one, two or three finger interaction). In the first case, we just have to wait for a short amount of time (500ms in the current implementation) before we assume that the user does not intend to use a multi-finger gesture. This proves problematic in cases where the user puts a finger on the table and starts to draw immediately (which is what all of the users in our evaluations did), while the actual drawing only starts after a short amount of time, ignoring the first 5 to 15 cm that the finger went from the initial touch position.

In order to circumvent this problem, we changed our code to start drawing immediately, but send a "cancel draw" event in case the cursor gets "upgraded" from a single-finger drawing cursor to a multi-finger move or delete gesture. Cancelled strokes are removed from the game world immediately.

With this approach, the user get instantaneous feedback about his actions without having to touch the surface with all fingers at the same time.

Another problem we faced was that the tracker software sometimes "lost" tracked fingers during a movement. In this case, a dragging event would normally be downgraded to a drawing event, and upgraded again as soon as the lost finger is available again.

To avoid such problems, the gesture recognition keeps all movements active as long as at least one of its fingers is reported by the tracker software. If a new tracked point appears near a gesture that lacks at least one point, it is "absorbed" and added to the gesture without causing a draw event itself.

3.5 The Multi-Touch Main Loop

```
# objects for TUIO and cursor tracking
tuio_input = tuio.Tracking('')
```

```
tracker = CursorTracker()

while True:
  while tuio_input.update():
    pass # read pending data from socket

  # send touch points to our tracker
  tracker.update(tuio_input.cursors())

  for event in tracker.get_events():
    # post new events created by tracker
    numptyphysics.post_event(event)

  while not tuio_input.update() and not tracker.has_events():
    tracker.idle() # wait for new touch events to happen
```

Listing 1: Simplified multi-touch main loop

The multi-touch input code runs in a separate thread from the normal, SDL-based main loop of the game. You can see a simplified version of the multi-touch input loop in listing 1. We “mask” the raw TUIO cursors by implementing a CursorTracker object that detects gestures and manages events.

This design allows for easy enhancement of the recognized gesture by simply adding new code to the CursorTracker implementation. It also allows us to replace the TUIO-based input with some other method. The event structure we use to communicate with the multi-touch input code and Numpty Physics carries no TUIO-related information, but only information relevant to cursor tracking.

3.6 C++/Python Event Interface

In order to send events from the Python-based multi-touch code to the C++-based game, we implemented a new Python module called “numptyphysics” in C++ which provides the necessary integer constants and methods to post events. The event object structure itself is written in Python, but C++ code is used to parse the data and convert it to C structs that are passed to the game code using pointers. A partial list of contents of the module in Python syntax can be seen in listing 2.

```
# Python module: numptyphysics
START_STROKE = (integer value of SDL_NP_START_STROKE)
```

```
APPEND_STROKE = ...
FINISH_STROKE = ...
...
WIDTH = (width of screen as integer)
HEIGHT = (height of screen as integer)
MAX_CURSORS = (integer value of MT_MAX_CURSORS)

# Method to post events to the main loop
def post_event(evt):
    (implemented in C++)
```

Listing 2: Public Interface of the numptyphysics module

Listing 3 shows the definition of an input event in Python syntax. As you can see, no TUIO-specific data is passed from the input module to the C++ code. The CursorTracker object keeps track of assigning cursor IDs to each event. Cursor IDs are integer values in the range (0, MAX_CURSORS) that Numpty Physics uses to distinguish between different events for different strokes and operation. As soon as one cursor ID is finished (e.g. the stroke has been drawn), the cursor ID can be re-used for another stroke.

The event type is an integer value taken from the constants defined in the numptyphysics module (see listing 2), and the X and Y attributes are obviously the coordinates of the touch point.

```
class InputEvent(object):
  def __init__(self, x, y, event_type, cursor_id=0):
    self.x = x
    self.y = y
    self.event_type = event_type
    self.cursor_id = cursor_id
```

Listing 3: Input event structure

3.7 Multiplayer Modes

Multi-touch devices inherently support collaborative multi-user applications, especially multiplayer games. This topic has been covered extensively in prior work (see Khaled et al, 2009; Gross et al, 2008; Hartman, 2008).

However, the employed hardware imposed some restrictions on the design of the new multiplayer modes because it does not provide user identification. This constraint was circumvented by doing an ad-hoc assignment of tokens to players: each player is implicitly responsible for the token next to him.

As Numpty Physics was not designed for multiple players initially, player information for tokens and goals was not contained in the level format. We extended the format to store the player number with each token and goal.

There are no restrictions on the allowed actions for each player. Everyone can draw everywhere and also move and delete the opponent's objects. This resulted in three multiplayer modes:

3.7.1 Single-Goal-Versus

In the Single-Goal-Versus mode the two players have their own token, which they have to bring towards one common goal item.

Levels for this mode contain one goal with number 0 and two tokens with different numbers.

The player who is the first to reach the goal with his token wins the level.

3.7.2 Multi-Goal-Versus

With Multi-Goal-Versus there are at least two goals, each belonging to a token.

Accordingly, this mode also uses two tokens with different numbers. Additionally there is at least one goal for each token-number.

The player who has collected his goals first wins the level.

3.7.3 Single-Goal-Cooperative

In the Single-Goal-Cooperative mode two or more players share a single token which they use to cooperatively reach one or more goals.

As no user information is needed for this mode, each of the existing single-player levels can be used.

3.8 Level Design

We created six new levels (figure 12) specifically designed to work best with our multi-touch version of the game. This was needed, as the original game did not feature multi-player levels. We also re-

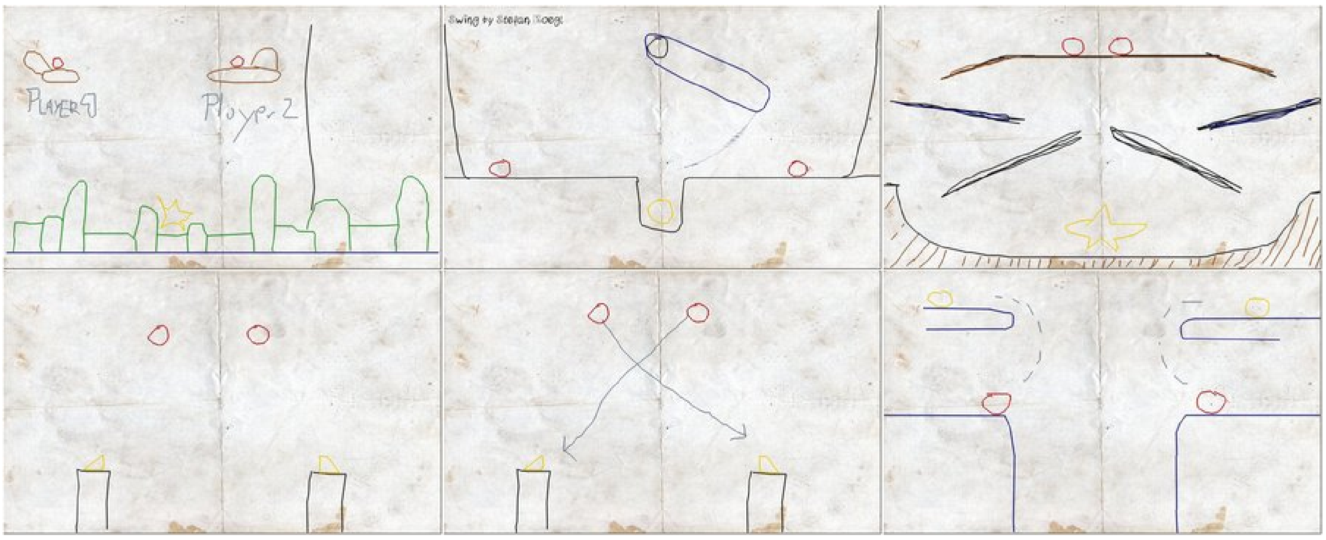

Figure 12: Six new multi-player levels we designed for Numpty Physics

designed some of our custom levels to change the area of interaction in order to work around several limitations of the hardware that we used (especially the top and right areas of the screen, as they had false blobs).

Multiplayer levels were designed to be fair, so most of them were symmetrically with the tokens on the left and right edges of the game world. Quick and competitive matches were used to require more engagement from the players, which we hoped would allow better evaluation. The levels were created using the existing level editor and then manually modified to the multiplayer format using a text editor.

As already pointed out, levels that did not take the hardware restrictions into account were not playable on the table. The third level in the first row of figure 12 (“Kugelbahn”) was unplayable by all of our testers due to the bad responsiveness of the table hardware in the upper area of the screen.

We also tried to modify some levels to work around the hardware problems we had in the first evaluation with some levels. As you can see in the first level in the first row of figure 12 (“Kaktus”), the starting point of player 2 was moved from the right-hand side of the screen towards the center. While in the first evaluation this level was not playable by some testers, the modified layout that was used in the second evaluation made this level playable by all testers due to the better responsiveness of the hardware.

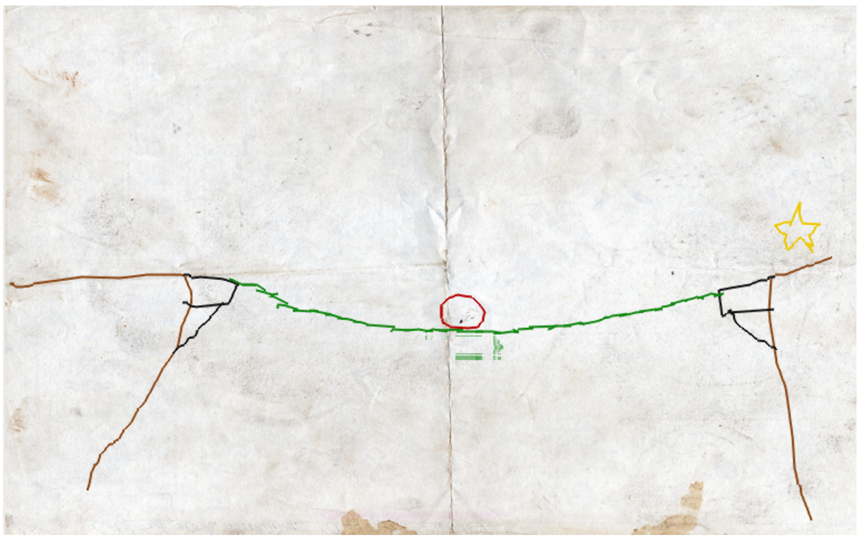

Figure 13: Level of the original Numpty Physics that uses a rope

3.9 Elastic Lines – Ropes

To further leverage the possibilities of multi-touch input, a new feature to draw elastic lines, also called "ropes" has been planned.

A rope can be used to bounce other shapes and has been used by the original game in one of the shipped levels, shown in figure 13. It was created using the level editor by joining many small regular lines. As creating such shapes in the regular game mode is not practical, we planned to implement an additional gesture which would create such a rope.

The feature has been implemented in the game code and the mouse handling code has been extended to allow using the feature using regular control mechanisms. As the multi-touch input has not been finished in time for the evaluations, this feature is not described here in detail.

3.10 Audio Output

In order to add more feedback to the user's input actions, we implemented support for playing back sound effects. A summary of events and the associated sound effects is shown in table 2.

According to prior research (Brewster and Crease, 1999), adding sound to graphical output can improve the usability of the interface. We think that in the case of a multi-touch table, good audible

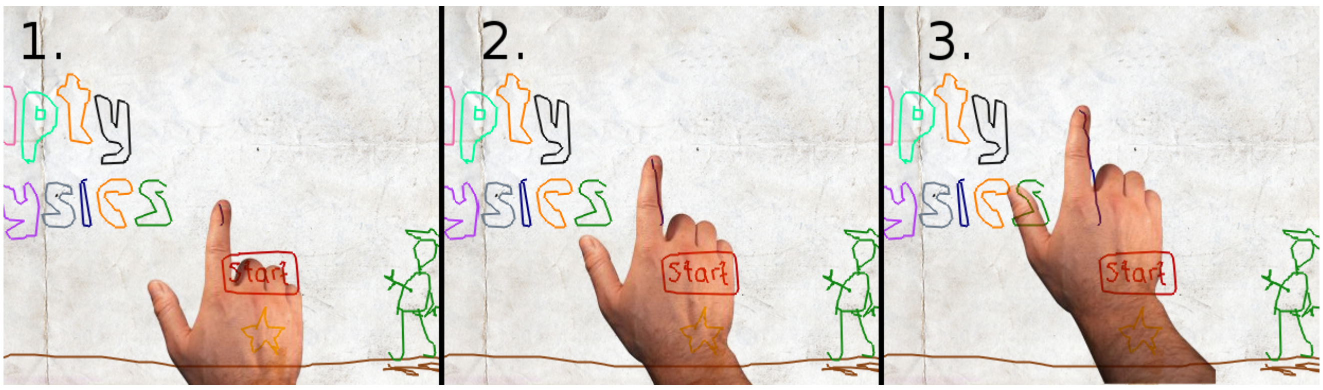

Figure 14: Example of the user's hand obscuring a line that is drawn

feedback will improve usability, because it can make up for some of the disadvantages of the technology: The hardware and tracking mechanism used introduce a short delay from the physical touch of the finger to the real tracking point being transmitted to the game engine. Also, the fingers and hands used to interact with the table sometimes obscure parts of the surface, making it hard to see strokes as they are drawn (see figure 14 for an example of such a movement in Numpty Physics).

Providing a constant sound loop while a drawing action is taking place allows the user to be confident in the stroke being drawn even when he cannot see the stroke, because it is obscured by the hand of the user. The instant muting of the sound when a stroke is finished also alerts the user that the drawing operation has been stopped.

Gärdenfors (2003) suggests that games based on sound can enhance user feedback. Even though Numpty Physics does not rely completely on sound, we considered some ideas from Gärdenfors (2003) when planning audio output.

We tested different sound effects with the game by adding the sound effect to the game and then playing a level and trying out all possible actions. For the drawing and moving actions, we picked loops that would play endlessly as long as the action has not been finished. The sounds were faded out as soon as the action has been finished. For deleting lines we decided to play a short sound of an explosion - one for each line deleted.

In order to make the sound effects and feedback more realistic and because we allowed multiple players in the game, the code was designed to allocate as many mixer channels as there were cursors. We also implemented a stereo effect for the sounds, so that drawing on the left edge of the table would

Event / action	Sound effect
Drawing a line	Chalk scribbling on a blackboard
Moving a line	Weakened sound of water bubbles
Deleting a line	Short sound of an explosion

Table 2: User actions and their associated sound effects

result in the drawing sound coming out from the left speaker only.

This stereo effect is especially important for multiplayer games where two players will simultaneously draw strokes. Without the stereo effect, the drawing sound would come out of both speakers at the same volume level, and the same feedback would be provided for both players regardless of who is currently interacting with the table. Using separate headphones for each player and only playing feedback for each player on their headphones is not an option, as the hardware we utilized does not allow for user identification, which is required for determining the headphones on which to play back the sound effect.

Of course, the stereo effect is tied to the X coordinate of the current action, so drawing a stroke from left to right will result in the sound fading from the left channel to the right channel. It is important to note that in order to really make use of the stereo effect, using stereo speakers and placing them in good positions on the table is crucial for the feedback to be effective.

We used the SDL_mixer library for playing back sounds and positioning them in stereo space using the touch coordinates of received input events. Sounds used were obtained from the freesound project (Music Technology Group, 2005), which provides sound effects and loops under a Creative Commons license.

You can see the stereo speakers we used in our evaluation and their placement on the multitouch table in figure 18.

3.11 Problems

There were several blockers that required us to spend special effort to fix in order to be able to continue development.

3.11.1 Mac OS X and Full-Screen Mode

The tracker application (tbeta) and some of the used web cams only worked well on Mac OS X (tbeta is available for Linux, Windows and Mac OS X, but camera support differs between operating systems).

Therefore, we decided early in the development process that we would carry out the evaluation and most of the development on Mac OS X. We still did some development on Linux, and also built the finished game for Windows, but Mac OS X was where most of the testing happened.

Mac OS X was not supported by the original game at the start of our project. Benjamin Milde has built a binary version of Numpty Physics for Mac OS X, but even after contacting him, we did not get a reply and the requested source code. We therefore ported Numpty Physics to Mac OS X and found a problem with the full-screen mode (the game always segfaulted when run in full screen mode).

Only after two hours of debugging, we found out that the original game did initialize the SDL video mode twice, which resulted in a segfault in Mac OS X, but not on other platforms (Linux and Windows). We fixed this problem so that we were able to carry out our evaluations in full-screen mode on Mac OS X and also integrated the fix in the upstream code repository, so that Mac OS X was officially supported by the Numpty Physics source code.

3.11.2 Lack of Hardware

As always with projects that depend on special hardware (a multi-touch table), development is difficult when the hardware on which the project is to be run is not available during development. We were allowed to use the table at the institute as often as needed, but some time still needed to be spent at home writing code.

We therefore built multi-touch input devices (MTmini) and used an iPod Touch with the OSCemote software to have multi-touch input at home. Due to the nature of our project, having this indirect way of input made it difficult to start strokes exactly where we wanted.

In order to overcome this problem, we added a preview functionality to the game which made it possible to draw small, black rectangles at all detected touch points, which made it a bit easier to check to which screen spot finger touches would map.

In addition to the indirect input method using the MTmini, touchscreen-based methods like the OSCemote software on iPod Touch devices provided a good and exact way of providing multi-touch input. The behavior of input points from the iPod Touch and the real multi-touch table tends to be different (there is more distortion and noise in the cursor movements). This resulted in some unexpected behavior when we tested our software (which worked flawlessly when tested using simulated multi-touch input) on the real multi-touch table.

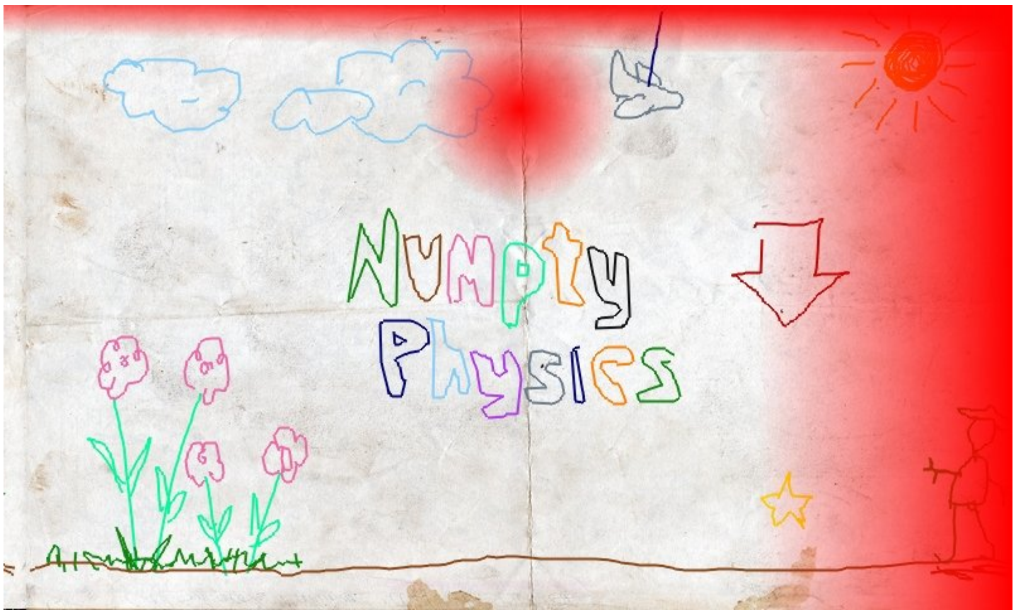

Figure 15: Unresponsive areas on the multi-touch table surface

As a consequence, we decided to spend an afternoon at the table to tailor our software to the multi-touch table's behavior before the second evaluation. This helped making the multi-touch input more robust and allowed us to carry out the evaluation with less problems than the first evaluation.

3.11.3 Table and Tracker

There have been some problems with the responsiveness of the multi-touch table that has been used to carry out the evaluation. As you can see in figure 15, the right-hand side of the screen was especially prone to responding badly or not at all. A small part of the top of the screen was also very unresponsive. The spot in the top middle area of the screen (see figure 15) is the light cone of the infrared lamp that is built into the screen. In the tbeta video input, this is shown as a bright white spot. Finger touches in this area are also not detected very well, especially with small fingers or low pressure.

The latest version of tbeta (Community Core Vision 1.2) that has been used in the last evaluation had a bug where it would constantly send touch points at coordinates (0, 0) to the TUIO application. As this resulted in very strange artifacts, we worked around this bug by simply ignoring all cursors that appeared at (0, 0), which was not a really useful touch point in any levels, anyway.

3.12 Upstream Integration

During the development major parts of the application had to be studied and reviewed to get an idea of the relevant parts of the application. Based on this research, decisions about where to implement a certain feature have been made. While working on the code we discovered numerous bugs in the original source code which were fixed during the development. Some fixes that we have made have been integrated directly in the upstream project.

Further we made sure to keep all changes modular to allow integration of the multi-touch features in the upstream code repository eventually.

4 Evaluation

This section follows the Common Industry Format (see Edmonds, 2003) for Usability Test Reports. We have modified and tailored some parts to accommodate the fact that we were testing a game where the subjective experience of the users (having a “good time”) is more important than measuring the efficiency and time of solving each level.

4.1 Test Objectives

The aim of our evaluations was to determine the usability of our multi-touch port and the degree to which users take advantage of multi-touch input. As we also added multi-player support, evaluating the feasibility of playing the game with more than one player was also an objective of our evaluations.

The product tested in the first evaluation has revision ID 378669aa in the version control repository. Revision ID 02db192d was the version used to carry out the second evaluation. The link to the Git repository where the code can be obtained can be found on our website (Perl and Kögl, 2009).

As we have spent many hours playing the game while developing it, carrying out an evaluation with users who have not played the game was an important goal in order to avoid bias as a result of the learning effect.

Both evaluations targeted different subsets of the final game. The first was set at a relatively early stage to test the multi-finger input, restricted to one action at a time. The second evaluation contained multiple gestures at a time, as well as the multiplayer modes.

The evaluations were scheduled after the corresponding parts of the game seemed stable enough to be evaluated. After that, a date was fixed and time-slots were selected.

4.2 Planning

Our initial plan was to carry out a pre-evaluation as well as two regular ones for the different stages of the implementation to be able to course-correct any usability issues that could arise during the evaluation:

1. Pre-evaluation - tests with the original game to find areas needing enhancements
2. First evaluation - finger interaction, no multi-touch gestures
3. Second evaluation - special multi-touch, multi-user levels and menu structure

The fact that the game features multiplayer modes that were to be evaluated allowed us to use the method Co-Discovery (Dumas and Redish, 1999). Two or more people carrying out an evaluation together are much more encouraged to speak about the tasks at hand than a single person. This ensures a more natural situation without needing to urge the subjects to talk about their experience.

For single-player scenarios, we asked our test subjects to use thinking-aloud.

4.3 Pre-evaluation

Before carrying out the evaluation of our multi-touch port of Numpty Physics, we loaded the original version of the game on a Maemo device and gave it to 4 persons in order to get some general feedback. This pre-evaluation survey was informal and was intended to get an idea of which areas we should focus on when planning and implementing our multi-touch enhancements.

Here are some example questions that we used on the pre-evaluation subjects:

- How do you like the game concept in general?
- Is the stylus-driven input method intuitive?
- Are you comfortable with the screen size?
- How accurate is the stylus-driven touchscreen input?
- Has input been detected in a wrong way on this device?

In general, users were comfortable with the size of the screen for single player operation, but during our investigations, bystanders wanted to give hints and point to positions on the screen where the player should draw a new stroke, which proved to be difficult given the small screen size and the hand-held nature of Maemo devices.

The stylus-driven input method was very intuitive, as it resembles the use of a pen for drawing on a sheet of paper. There have been some slight problems with some long strokes not being detected correctly as a single stroke, which resulted in parts of the strokes "falling down" while the drawing operation was still in progress.

In general, people liked the concept of the game, and given our impressions of multiple bystanders watching a person playing the game, adding multiplayer abilities to the game would improve user learning and interaction between different players. This is in line with the findings of Rick et al (2009)

Evaluation	Pre-Eval	First (25.5.09)	Second (28.7.09)
Male	1	5	2 (1 new)
Female	3	5	2 (1 new)
Total	4	10	4 (2 new)

Table 3: Number of test subjects used in evaluations

where researchers found that "collaborative activity is generally beneficial to children's learning and development". While Rick et al (2009) focused specifically on children, our multi-touch port of Numpty Physics is suitable for users at any age.

Given the educational value of Numpty Physics for learning physics in a playful way our version of Numpty Physics will enable schools to use Numpty Physics as an interactive and playful way of learning about gravity while still being able to facilitate cooperation between pupils on a multi-touch table (Rick et al, 2009).

4.4 Method

In order to get representative results for our usability tests, we planned carrying out three evaluation runs, of which one had to be canceled because of calibration issues with the hardware.

4.4.1 Participants

Invitations were sent out to people that previously showed interest in the project. They were asked to select one of the available time slots, for which two people could register. This should ensure to have enough players for the multiplayer levels. Time slots were 20 minutes long with 10 minutes between evaluations to allow for recreation and any setup that needed to be done.

We used 10 subjects for the first successful evaluation and 4 subjects for the final evaluation. Table 3 lists the test subject distribution in more detail.

The test subjects were 19-24 year old students from Vienna in their second to fifth semester. All test subjects have used computers for at least 5 years and 4 test subjects worked or studied in the IT sector. None of the participants in the first evaluation had product experience with Numpty Physics.

In the second evaluation, we invited two participants (one male, one female) from the first evaluation and paired each of them with one new (no product experience) user to create mixed-experience teams for multi-player games.

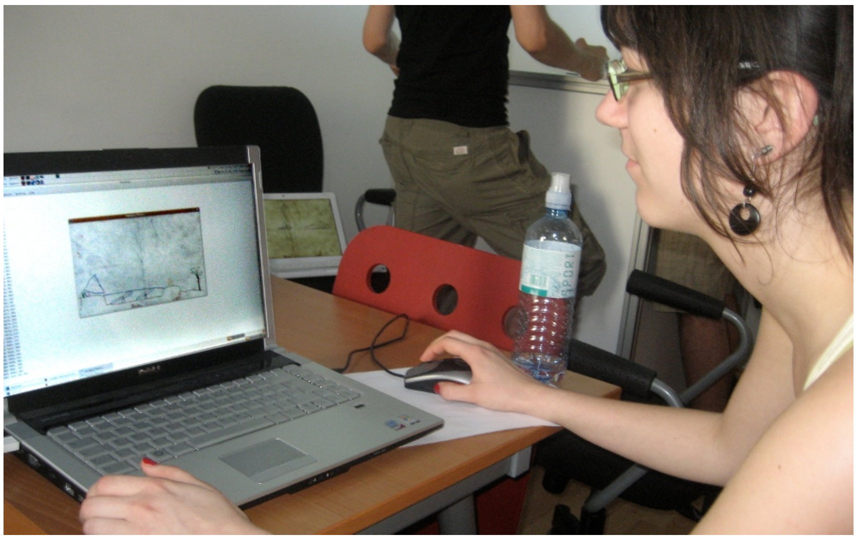

Figure 16: Test subject playing the original game on a notebook

4.4.2 Context of Product Use

Using the product was evaluated in three steps:

1. Playing the original Numpty Physics on a laptop using mouse input
2. Playing Multi-touch Numpty Physics in single-player mode on the table
3. Playing cooperative and versus multiplayer Numpty Physics on the table

The subjects were told to use the thinking-aloud method while playing the game and we logged all the issues and opinions that users had. In multi-player scenarios, users were allowed to help each other using the co-discovery method.

We asked the players to try out the original Numpty Physics on a notebook (as seen in figure 16) before switching to the multi-touch version, because the people should get a feeling about the existing version of the game. Based on this experience discussing the advantages of the multi-touch input was possible later on.

The next step was to play some single-player levels to get used to the new environment. The players could best experiment with the new kind of interaction without having another player to worry about.

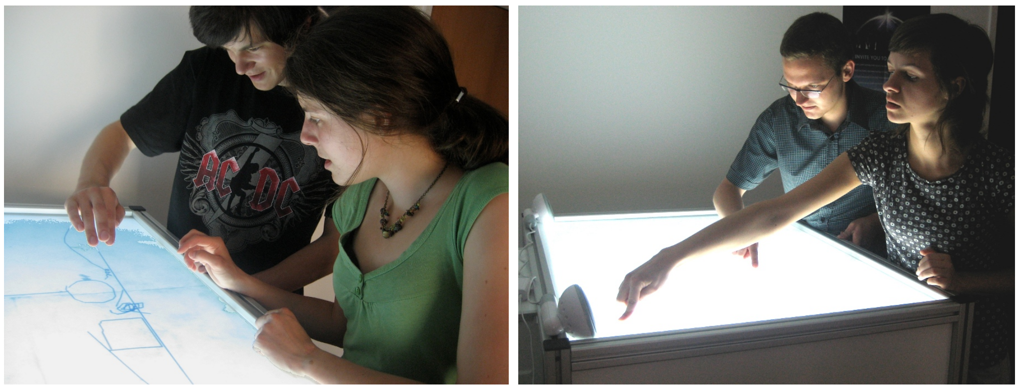

Figure 17: Subjects playing cooperative (left) and versus (right) games on the multi-touch table

The number of single-player levels depended on the advancements the players showed during the play. The switch to the multi-player levels were done when the player seemed practiced enough.

Playing the multi-player levels was the main part of the evaluation. The levels were especially designed for these tests to include all kinds of interactions and strategies. You can see test persons playing the game on the multi-touch table in figure 17.

The discussions about the game were informal small-talks about the users' experience on the table. All our test subjects have had their first contact with multi-touch tables during our evaluation, so most of the users were generally excited about our project.

4.4.3 Test Setup and Computing Environment

For reasons of easier debugging and observing, we set up our evaluation environment so that the evaluator was sitting with a laptop computer on the left side of the multi-touch table, as seen in figure 18.

We used a MacBook (Mid-2006) with a FireWire 400 port and a 2 GHz Intel Core Duo CPU and 2 GB of RAM to carry out the evaluations. The operating system used was Mac OS X 10.6 with all the latest patches and updates installed as of May 2009 (first evaluation) and July 2009 (second evaluation).

The software stack used was SDL 1.2.13, SDL_mixer 1.2.8, SDL_ttf 2.0.9, SDL_image 1.2.8 and Python 2.5. For PyTUIO, we always used the latest version available from the Subversion repository of the project at the time of the evaluations.

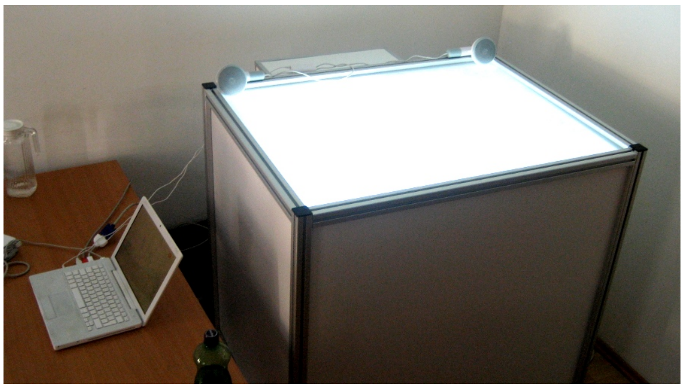

Figure 18: Hardware setup and environment used during the evaluation

For connecting the multi-touch table to our computer, we used a VGA cable to connect the projector (run at 1024x768 resolution) and a FireWire cable to connect the camera used for tracking touch points. The operating system was configured to mirror the 1024x768 image on both the VGA port and the computer's screen to help the evaluator observe a running game by watching the gameplay actions on the computer screen.

No audio device has been used in the first evaluation. For the second evaluation, we placed stereo speakers at the top edge of the multi-touch table (see figure 18) and set the volume to maximum in Mac OS X.

The command line used to start the game for evaluation purposes is seen in listing 4.

```
./numptyphysics --fullscreen --geometry 1024x768
```

Listing 4: Command line used to start Numpty Physics in evaluations

For generating tracking point logs, the file name of the level was appended to the command line above (which made the game jump directly to the level that has been specified). As point log data was saved on a per-session basis, Numpty Physics was closed down and re-started after a level has been completed and the resulting point log file has been renamed accordingly. Converting point log files to images for analysis and overlay is described in the results chapter.

4.4.4 Experimental Design

This section describes the test protocol and how the evaluations were carried out.

First, the participants were welcomed at the institute when they arrived at the start of their time slot and brought into the DECO.lab. When participants arrived earlier than expected, they were asked to wait outside the lab.

After welcoming the participants, they were briefed on the purpose of the evaluation and it was explained that the aim of the evaluation is to test the usability of our application, not their abilities to finish the game's levels.

All test subjects were told about their rights as human subjects as described in American Psychological Association (1982).

The participants then filled out a form with questions about their age, education, occupation, work experience, computer experience and if they have played Numpty Physics before. See the section about participants for detailed information.

After the general questionnaire, the product use was tested in three steps as outlined in the section "Context of Product Use". Users were encouraged to try out several approaches to solving a level before asking us for help.

Following the product use, we sat down with the participants and discussed all three variations (mouse-based, single-player and multi-player) and noted any comments and problems that the test subjects mentioned.

Some participants wanted to observe the evaluation following their own, which we allowed, although we asked observers to not say or comment on anything and to remain in a distance from the active participants in order to not interfere or influence the results of the evaluations.

Other than the test subjects themselves, only the two persons of the evaluation team were in the room while the evaluation took place. These two persons were also the ones who interacted with the test participants. Obviously, the test participants collaborated on their solution in multi-player games.

The participants were not paid or otherwise compensated.

4.4.5 Task Instructions

As the game itself was self-explanatory, we decided to not give formal task instructions to the participants in order to evaluate the obviousness of our UI. The first level of the game features an in-game description what has to be done in order to solve the level. A screenshot of this level can be seen in figure 19. This in-game method of describing how to play the game is also important for situations where Numpty Physics could possibly be presented in a public installation or a club where instructors

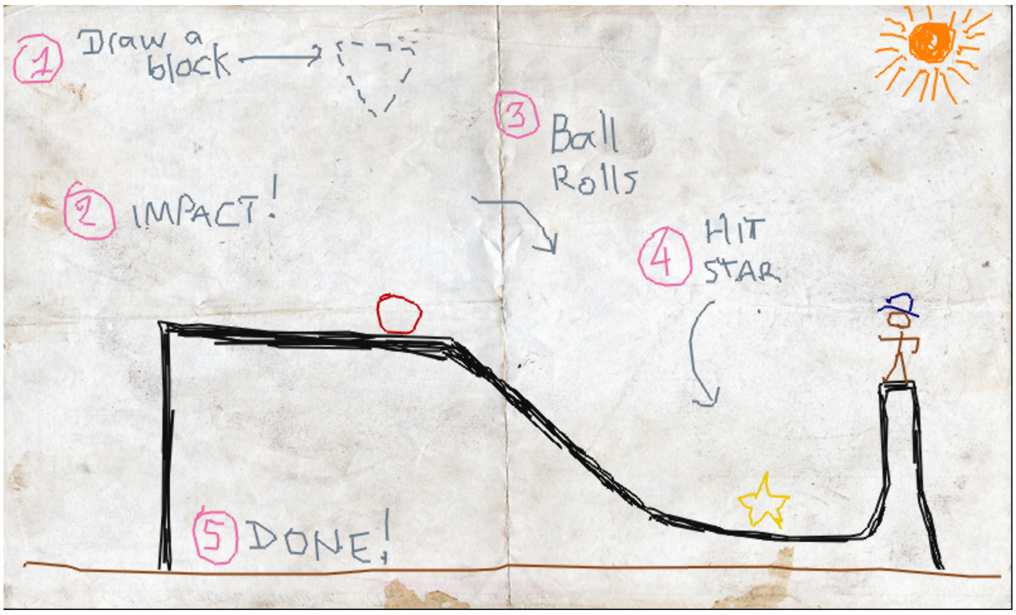

Figure 19: The tutorial level of Numpty Physics with general instructions

are generally not available to describe the gameplay to users coming to the table.

4.4.6 Draw-Only Mode

To avoid problems with mis-interpreted gestures, we implemented an additional game mode which only allowed drawing shapes, and disabled moving and deleting. This mode was used in the evaluations as needed, when it was obvious that the game did no produce the intended actions.

The draw-only mode is implemented completely in the gesture recognition code. When turned on it interprets every tracked point as a drawing actions. Two points near to each other are thus reported as two separate drawing actions, instead of one moving-action to the game. The game code itself is not aware of this mode, as it simply never receives drawing and deleting actions.

Before using this mode, the participants were told about the reduced features and the intentions of this setting. At first most of them still tried to move and delete shapes because they got used to it during the previous levels. After a short learning period most participants showed more satisfaction with the results of the game.

However sometimes the progress of the game required moving or deleting a certain stroke in order to finish the level. In this cases one of the evaluaters used the laptop computer beside the multi-touch table (see figure 18) to assist the players.

From the experiences gained from the draw-only mode we concluded that more time for adjusting the parameters of the gesture recognition would be needed. As this is strongly dependant on the used hardware and requires several further evaluation sessions and development time, this could be part of a future project.

4.5 Usability Metrics

Usability evaluations usually contain metrics such as effectiveness and efficiency. As we're dealing with the evaluation of a game here, we decided to use user satisfaction as the sole metric. This way, the "time spent well" is our measurement of Numpty Physics' usability. We utilized the Thinking Aloud method and note taking to capture opinions and thoughts of users to determine their satisfaction with the game.

4.6 Note Taking

Note-taking was done independently by both team members to ensure a more thorough view on the results. The notes were digitized after the evaluation, merged and grouped. Here are some comments from both evaluations that we have been able to gather (translated into English - the evaluation was held in German):

- "I have to press really hard to get my finger recognized"
- "It does not react to my finger presses anymore"
- "The stroke does not fall down right away, it pauses a bit"
- "I cannot give the object momentum by dragging it"
- "Why isn't there some kind of score after the end of the game?"
- "It seems like drawing with the thumb works best"
- "Why can I drag the menu with 1 finger when dragging is done with 2 fingers in the game?"
- "The sound for deleting an object should only be played when an object is really deleted."

- “This sounds as if the level is located underwater” (about the sound used for dragging objects)

Users were mostly complaining about the game in areas where the behavior did not match the expected behavior in a real-world scenario (the best example for this is the fact that dragging does not add/cause momentum to the dragged object - although we have done this deliberately to make the game harder and stop people from drawing objects and throwing them against their token).

Other problems like the fact that the game gets stuck once in a while are a result of the machine where both the tracker application and the game get stalled for a short amount of time when the video stream from the used FireWire camera gets interrupted. These problems can be overcome by using better and more capable hardware for processing the video input or by running the tracker and the game on separate machines.

In addition to analyzing comments and actions from users, the logged tracking points were used to analyze typical user behavior by creating images out of the log data and overlaying the resulting images over screenshots of the corresponding levels, as already discussed in the evaluation section.

4.7 Results

The general acceptance of the multi-touch version of Numpty Physics was very high. The new control mechanisms were described as intuitive by the test subjects.

4.7.1 Data Analysis - Logging Tracked Points

In order to get a more objective view of the areas on the table that are touched while playing a game, we implemented a logging facility in the multi-touch input code that logs the X and Y coordinates of the touch point and the cursor ID that is passed to the game. The timestamp of the touch point is also logged to help reconstruct the gameplay scenario as seen in listing 6.

```
import sys

ORIGINAL, TABLE = (800, 480), (1024, 768)
STYLE = 'fill:#000000;opacity:.2;'

cx = lambda x: int(x)*ORIGINAL[0]/TABLE[0]
cy = lambda y: int(y)*ORIGINAL[1]/TABLE[1]

print '<?xml version="1.0" encoding="UTF-8" standalone="no"?>'
print ('<svg xmlns="http://www.w3.org/2000/svg" width="%d" '+
        'height="%d" version="1.1">') % ORIGINAL

for line in sys.stdin:
    line = line.strip().split('/')
    if len(line) != 4:
        continue
    x, y, curid, timestamp = line
    print ('<path style="%s" d="m %d,%d a 5.5,5.5 0 1 1'+
            '-11,0 5.5,5.5 0 1 1 11,0 z"/>') % (STYLE, cx(x), cy(y))

print '</svg>'
```

Listing 5: Python script to convert pointlog to SVG

The simple format allows us to post-process the touch input data and analyze the gameplay and detect ”dead spots” on the surface (we used the code in listing 5 to generate SVG files). It also allows us to determine the areas of the screen that are mostly used in levels. This is especially important for multi-player levels where users should generally not have to use the same parts of the surface at the same time.

```
    479/346/8/1248795361
    209/547/7/1248795361
    479/350/8/1248795362
    218/551/7/1248795362
```

Listing 6: Example pointlog contents (X/Y/ID/TIME)

As you can see in figure 20, the level design for the multi-player “Swing” level (a level that we created specifically for the versus multi-player mode) divides the touch area in two halves where most of the interaction takes place. There are just a few touch points in the middle of the game area, where a horizontal line was drawn.

Looking at the spikes coming out from the bottom of the touch-point cloud in the left half of the screen (labeled “1.” in figure 20), we can see that the user playing this level on the left side of the table mainly used falling triangular objects to move his token forward. Another possible strategy to solve this level would be to draw a big triangular ramp and use falling objects to give momentum to the user token (an example where this strategy was used can be seen in figure 22).

The dark spot on the ground in the right half of the screen (labeled “2.” in figure 20) took place later in the game and has been a delete stroke action.

A different example for a multi-player level created by us is the “Rope” level. Like the “Swing” level, “Rope” has a symmetric appearance. As you can see in figure 21, most of the touch points appear in the middle third of the screen (labeled “1.” in figure 21). Only a small percentage of the touch points appear on the left and right sides of the game area, suggesting that hands of the users might collide when this level is being played.

In fact, test subjects reported problems with this level due to the limited screen space for two users operating at the same time on the same area.

This could be fixed by using bigger hardware (which would introduce other problems) or by simply designing all multi-player levels so that the main touch areas are different between players. The “Swing” level shown in figure 20 is a good example of a well-designed level providing good physical

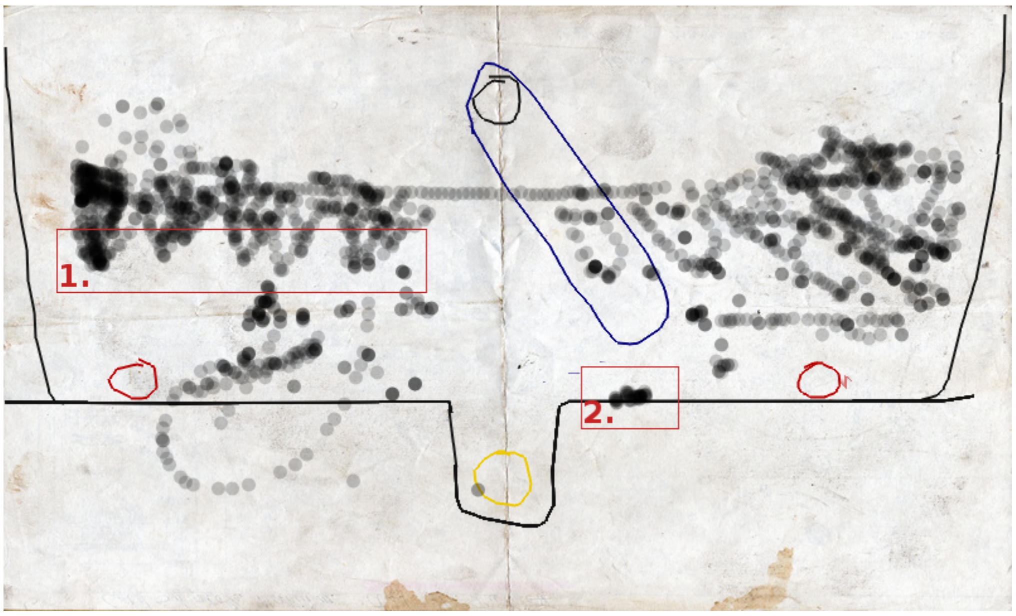

Figure 20: Touch points for the “Swing” level (second evaluation)

separation between the hands of two players.

Another noteworthy fact that you can deduct from figure 21 is the heavy use of attaching strokes to non-moving ground objects. In the screenshot, you see multiple diagonal lines of touch points that go through the left and right ground objects (labeled “2.” and “3.” in figure 21). Drawing strokes through other objects results in the strokes joining at the intersection point. This can be used to prevent objects falling down from falling into the pit in the bottom middle of the level.

To compare the touch interactions of our new multi-player levels with existing single-player levels from the original game, we chose to analyze a sample game of “Plane Sailing”, the third level of Numpty Physics (see figure 22 for a screenshot with touch points).

This level has been played by two players, but only one player drew on the table at a time on a turn-by-turn basis (the other player was observing the game and thinking about the next step during that time).

At first, the players drew a triangular shape (marked by the rectangle labeled “1.” in figure 22) and then drew a filled ball (the black spot on the left side, labeled “2.”) that rolled down the slope that has been created by the triangle. After that, several diagonal lines (one of them has been marked

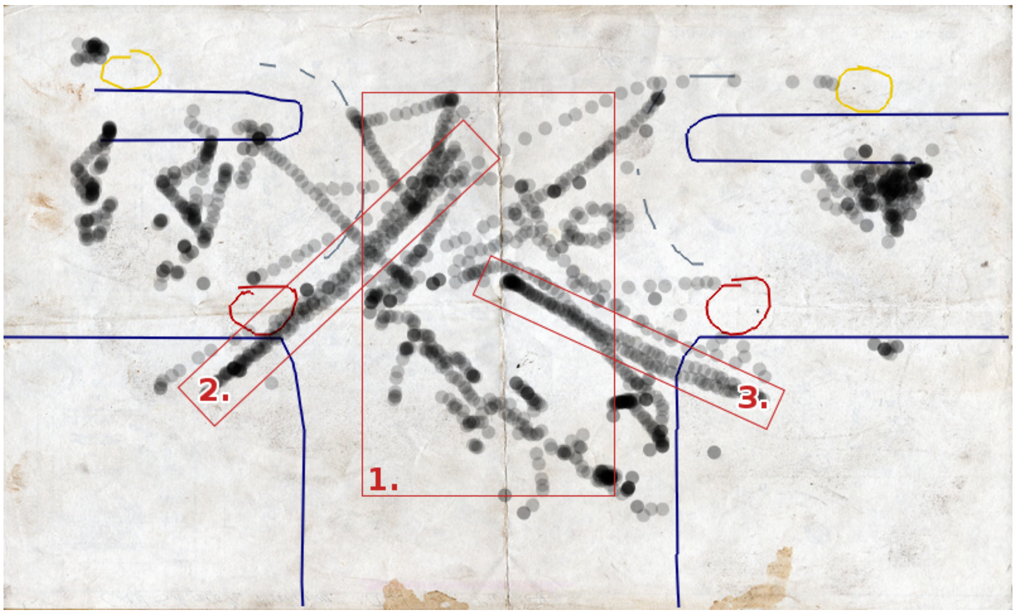

Figure 21: Touch points for the "Rope" level (second evaluation)

as "3." in figure 22) were used to draw a lever that would push the token towards the goal.

The single touch points scattered throughout the screen that you see in figure 22 are false blobs reported by the tracker.

In the case of false blobs, we modified the code to avoid having bogus strokes that appear as debris falling down in the game. The modification checks the number of points used in a stroke, and if this number goes below a pre-defined level, the stroke is discarded. In our tests, this did not cause strokes that users deliberately drew to disappear.

4.7.2 User Experience

We observed that the subjects were initially enthusiastic because of the appearance of the multi-touch table, which could have had a positive effect on the subjective user experience.

Drawing actions were carried out more accurately on the multi-touch table compared to control using the mouse. We assume this is a result of interacting with the table directly as opposed to the indirect input method provided by a computer mouse. Still, multi-touch drawing on the table was not as precise as stylus-driven input on tablets due to hardware constraints.

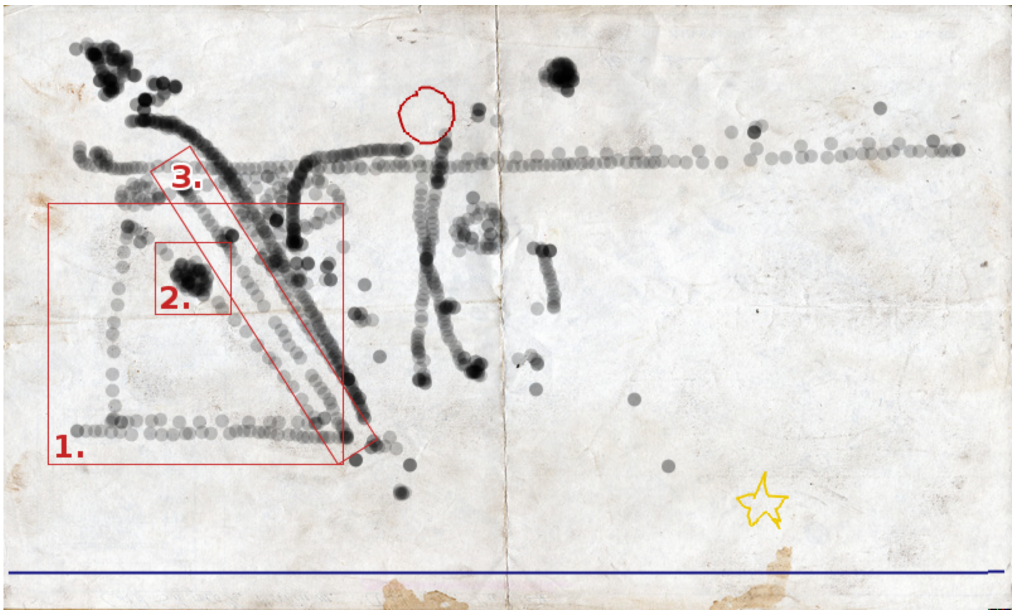

Figure 22: Touch points for the “Plane Sailing” level (second evaluation)

With multi-touch finger input, users expected different possibilities of interacting with the game. When moving objects, some of the users tried to throw them. We assume that because of the more natural control of the game, this movement seemed obvious to the users. Throwing objects is not possible in our implementation for fairness reasons in multi-player games and because the original game also did not allow throwing of objects.

Some testers also tried to scale objects, which does not generally constitute a natural movement, but probably was known from other multi-touch applications seen on videos on the web.

The aforementioned interaction mechanisms were not present in the game during the evaluation and would therefore be a good starting point for further enhancements to the game.

4.7.3 Technical Issues

As the input mechanisms were mostly tested using a MTmini and OSCemote on an iPod touch, input handling was optimized for these devices. This resulted in not well-tuned settings for gesture-recognition, which were partially not discovered before the evaluations.

The first version which was to be evaluated used a short delay for allowing all fingers of a gesture to be placed on the table and be recognized by the software. This delay was perceived as annoying by the players because it slowed down the whole game and made drawing small objects nearly impossible. This problem was progressively reduced in the consecutive versions by introducing a preview functions that outlined to-be-drawn lines while waiting for further fingers to arrive on the screen.

Additionally, inaccuracies of the used hardware made some regions of the surface virtually unusable, the subjects tended to continue playing while concentrating their play on working regions.

4.7.4 Single-Player Improvements

In addition to the direct interaction for drawing strokes in the original Numpty Physics, our finger gesture recognition code adds support for deleting and moving strokes by directly interacting with the objects to be affected. In the original game, this could only be accomplished by using a stylus on the single-touch screen in combination with hardware buttons that had to be pressed during the move or delete operation, or with different mouse buttons in the case of the Desktop version of Numpty Physics.

We initially expected the possibility for a single player to draw strokes with two hands would lead to interesting gameplay situations, however we have found in our evaluations that people are not able to draw strokes with two hands simultaneously on a multi-touch table (just as drawing with both hands on a sheet of paper is very difficult and does not feel natural).

There were no problems with letting multiple players cooperate on the default single-player levels that are installed with the original Numpty Physics.

4.7.5 Multi-Player Mode

The multi-player mode was well accepted by the players. Even after explicitly starting a single-player level, other players standing at the table joined the primary player giving advice and assistance directly in the game. Such social interaction seemed to have a positive effect on the user experience.

Most of the multi-player levels started with two tokens on the far left and right sides. This was intended to allow more space for the play. Because the table's motion tracking was not very accurate on the borders, these levels were frustrating for the players, because it was hard to move the token to the middle of the screen where the tracking was more accurate.

During some multi-player games, tokens were tossed out of the screen either deliberately or by accident. The corresponding player could then further participate in the game but was no longer able to win the level. This problem could simply be solved by respawning the respective object once it is off-screen.

Although the winning player is shown after each level, users requested statistics that summed up the players results after multiple levels.

5 Conclusions

Based on the results of our evaluations, we conclude that multi-touch has greatly enhanced the user experience of Numpty Physics. We suggest that the game itself was very appropriate for this kind of experiment, because it utilizes very simple and natural gestures and the actions carried out by the user are not very abstract.

Furthermore the game uses very little (implicit) rules which could constrain the port to multi-touch input. Numpty Physics does not require particularly precise input, which made it a good choice for multi-touch tables.

Our game generally does not have a single correct way of solving a given level – it allows multiple creative ways of doing so. This property is in line with the inherently natural multi-touch input.

Major problems faced during the evaluations were caused by errors in the code and solved until the following user evaluation or constraints imposed by the hardware used. Such errors include "blind" spots on the surface and both false-positive and false-negative tracked points. These restrictions proved to be annoying for players, especially in multiplayer games against each other.

We found that the differences in thickness of player fingers as well as the pressure applied onto the table were contributing factors to touches not being detected correctly.

Simple tests carried out with more professional hardware did not show such problems. Better hardware would therefore allow smoother gameplay.

These problems were reduced by a feature of the gesture recognition code that allowed fingers from a multi-finger gesture to disappear and reappear later. This was also the source of related problems because separate actions were sometimes grouped together based on their physical distance. In order to make good use of this feature, the settings for this feature should be fine-tuned for the size of the multi-touch table that is being used.

Issues with feedback were then further reduced by providing audio output during multi-touch actions. It signaled the players – especially the more advanced computer users – why an action was not carried out as intended. This feature could also be used with proper hardware to communicate other kinds of errors, such as a delete gesture without an object to be deleted below the finger.

The multiplayer mode proved to be a great enhancement to the overall user experience. The multi-touch table allowed this kind of interaction due to its physical size. Multiplayer games were not possible using a mouse-based desktop computer. Multiplayer modes were considered more fun than the corresponding singleplayer modes by our test subjects.

Social interaction during the game played a crucial role, especially in the cooperative mode where users had to coordinate their actions.

5.1 Summary

Multi-touch input is a good way to improve the user experience for certain kinds of existing games. This applicability depends on the interactions already available in the game. Providing suitable feedback, either visually or audible (or both) is crucial for the success of such an undertaking.

When planning to port an existing game to multi-touch input, the possibilities of multiplayer modes, both cooperative and versus, should be considered carefully.

Limitations imposed by the hardware, like possible inaccuracies and the lack user of user identification should also be taken into account.

6 Further Work

Based on the results of the evaluation and related research we present some ideas and concepts for further development and studies.

Building upon the research described in Wang and Ren (2009) utilizing physical properties of finger input would allow for more natural interaction. Numpty Physics currently allows for different types of shapes for level background and decoration, but only one kind of shape can be selected by the user. Taking input properties such as contact area and orientation into account would allow the user to differentiate types of strokes by finger orientation.

Another area of improvement would be to assign more physical weight to strokes that were drawn with high pressure. Unfortunately pressure-sensitivity is a feature that many DIY multi-touch tables currently lack due to the design of the hardware, so we were not able to evaluate the results of adding pressure sensitivity to the game.

Visual feedback could be incorporated into the game for dragging strokes and deleting strokes. Existing strokes could be decorated with a glow effect while they are being dragged. Similarly deleted strokes could be animated to either explode or shrink and then disappear instead of just disappearing. The real benefit of visual feedback should be evaluated after the feature has been implemented to confirm the expected gain in usability.

Further features could be implementing that build upon the multi-touch input mechanisms, such as the rope input described in section 3.9. This generally requires work in most parts of the game. The rope-feature has already been implemented in the game code, including event handling. It still needs implementation in the gesture recognition and additional levels using the new functionality.

In terms of audible feedback, our current stereo output (with left and right speakers) could be enhanced to a four-channel surround system, placing one speaker in each corner of the table. The relative volume for each speaker could then be determined by the normalized euclidean distance of the touch point to the corner where the speaker is positioned. Research in the area of sound-based game design (Gärdenfors, 2003) suggests that even surround sound does not provide enough positional data for simultaneously mixed audio samples, so the effectiveness of this approach would have to be tested in an evaluation comparing different sound output methods.

When implementing new means of interactions, the corresponding gestures have to be chosen with respect to existing ones, so that they can be differentiated by the application correctly. Nacenta et al (2009) compares several ways of separating gestures for scaling, rotating and moving while using two fingers for all of them. They evaluate different strategies and discuss them regarding various aspects.

In general, the limiting factor for Numpty Physics on a Multi-touch table was the inaccuracy of the hardware input, which needed some workarounds in software. These workarounds introduced some delays in the game's reaction to user input. Further improvements of the game in terms of speed and accuracy need a more robust hardware design for multi-touch tables. For the current generation of DIY multi-touch hardware, the ability to move and delete strokes inside the game makes up for some deficiencies of the hardware.

7 Acknowledgements

We would like to thank Tim Edmonds for creating Numpty Physics and releasing it as Free Software.

Further we thank Stefan Bachl, Stefan Kuschnigg and Christoph Wimmer from the research group for Industrial Software at the Vienna University of Technology for supporting this project in the context of a seminar on Human Computer Interaction.

We would also like to thank everybody who participated in evaluating Multi Touch Numpty Physics in both evaluation sessions.

8 Licensing

[4]http://creativecommons.org/licenses/by-nc-nd/3.0/

9 References

[American Psychological Association 1982] American Psychological Association: *Ethical Principles in the Conduct of Research With Human Participants.* 1982

[Brewster and Crease 1999] Brewster, Stephen A. ; Crease, Murray G.: Correcting Menu Usability Problems With Sound. In: *Behaviour and Information Technology* (1999), Nr. 18, p. 165–177. – URL http://www.dcs.gla.ac.uk/~stephen/papers/BIT99.pdf. – date visited: 2003-11-24. ISBN 0144-929X

[Buxton 2007] Buxton, Bill: Multi-Touch Systems that I Have Known and Loved. (2007). – URL http://www.billbuxton.com/multitouchOverview.html

[Catto 2007] Catto, Erin: *Box2D Engine.* 2007. – URL http://www.box2d.org/. – date visited: 2009-08-13

[Dumas and Redish 1999] Dumas, Joseph F. ; Redish, Janice C.: *A Practical Guide to Usability Testing.* Intellect Books, 1999. – URL http://books.google.at/books?id=4lge5k_F9EwC&lpg=PP1&pg=PA31. – date visited: 2009-08-18. – ISBN 1841500208

[Edmonds 2003] Edmonds, Andy: *Common Industry Format Template.* 2003. – URL http://uzilla.net/uzilla/blog/cif/template.html. – date visited: 2009-08-12

[Edmonds 2008] Edmonds, Tim: *Numpty Physics.* 2008. – URL http://numptyphysics.garage.maemo.org/. – date visited: 2009-08-12

[Gross et al 2008] Gross, Tom ; Fetter, Marko ; Liebsch, Sascha: The cueTable: Cooperative and Competitive Multi-Touch Interaction on a Tabletop, 2008

[Gärdenfors 2003] Gärdenfors, Dan: Designing sound-based computer games. In: *Digital Creativity* (2003), Nr. 14, p. 111–114. – URL http://www.informaworld.com/smpp/content~db=all~content=a725290461. – date visited: 2009-08-12

[Han 2005] Han, Jefferson Y.: 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.* New York, NY, USA : ACM, 2005, p. 115–118. – ISBN 1-59593-271-2

[Hartman 2008] Hartman, Hans: *Multiplayer games and physics on multi-touch screen devices.* 2008. – URL http://epubl.ltu.se/1404-5494/2008/017/. – date visited: 2009-08-13

[Kaltenbrunner et al 2005] Kaltenbrunner, Martin ; Bovermann, Till ; Bencina, Ross ; Constanza, Enrico: TUIO: A Protocol for Table-Top Tangible User Interfaces. (2005). – URL http://modin.yuri.at/publications/tuio_gw2005.pdf

[Khaled et al 2009] Khaled, Rilla ; Barr, Pippin ; Johnston, Hannah ; Biddle, Robert: Let's clean up this mess: exploring multi-touch collaborative play. In: *CHI EA '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems.* New York, NY, USA : ACM, 2009, p. 4441–4446. – ISBN 978-1-60558-247-4

[Leidel 2007] Leidel, Jannis: *pytuio.* 2007. – URL http://code.google.com/p/pytuio/. – date visited: 2009-08-12

[Matejka et al 2009] Matejka, Justin ; Grossman, Tovi ; Lo, Jessica ; Fitzmaurice, George: The design and evaluation of multi-finger mouse emulation techniques. In: *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems.* New York, NY, USA : ACM, 2009, p. 1073–1082. – ISBN 978-1-60558-246-7

[Mehta 1982] Mehta, Nimish: *A Flexible Machine Interface*, University of Toronto, Master thesis, 1982

[Multitouch Barcelona 2008] Multitouch Barcelona: *Multitouch Crayon Physics.* 2008. – URL http://blog.multitouch-barcelona.com/2008/05/multitouch-crayon-physics-is-available.html. – date visited: 2009-08-12

[Music Technology Group 2005] Music Technology Group: *the freesound project.* 2005. – URL http://www.freesound.org/. – date visited: 2009-08-13

[Nacenta et al 2009] Nacenta, Miguel A. ; Baudisch, Patrick ; Benko, Hrvoje ; Wilson, Andy: Separability of spatial manipulations in multi-touch interfaces. In: *GI '09: Proceedings of Graphics Interface 2009.* Toronto, Ont., Canada, Canada : Canadian Information Processing Society, 2009, p. 175–182. – ISBN 978-1-56881-470-4

[Nakatani and Rohrlich 1983] Nakatani, Lloyd H. ; Rohrlich, John A.: Soft machines: A philosophy of user-computer interface design. In: *CHI '83: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. New York, NY, USA : ACM, 1983, p. 19–23. – ISBN 0-89791-121-0

[NUI Group 2009a] NUI Group: *Diffused Illumination*. 2009. – URL `http://wiki.nuigroup.com/Diffused_Illumination`

[NUI Group 2009b] NUI Group: *tbeta*. 2009. – URL `http://tbeta.nuigroup.com/`. – date visited: 2009-08-12

[Perl and Kögl 2009] Perl, Thomas ; Kögl, Stefan: *Multi-Touch Numpty Physics Website*. 2009. – URL `http://thpinfo.com/2009/mt`. – date visited: 2009-08-12

[Rick et al 2009] Rick, Jochen ; Harris, Amanda ; Marshall, Paul ; Fleck, Rowanne ; Yuill, Nicola ; Rogers, Yvonne: Children designing together on a multi-touch tabletop: an analysis of spatial orientation and user interactions. In: *IDC '09: Proceedings of the 8th International Conference on Interaction Design and Children*. New York, NY, USA : ACM, 2009, p. 106–114. – ISBN 978-1-60558-395-2

[Sandler 2008] Sandler, Seth: *MTmini - DIY Multitouch Mini Pad*. 2008. – URL `http://sethsandler.com/multitouch/mtmini/`. – date visited: 2009-08-12

[Wang and Ren 2009] Wang, Feng ; Ren, Xiangshi: Empirical evaluation for finger input properties in multi-touch interaction. In: *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*. New York, NY, USA : ACM, 2009, p. 1063–1072. – ISBN 978-1-60558-246-7