TabletopCars - Interaction with Active Tangible Remote Controlled Cars

Chi Tai Dang
Augsburg University
Department of Computer Science
Human Centered Multimedia
Universitaetsstr. 6a
86159 Augsburg, Germany
dang@informatik.uni-augsburg.de

Elisabeth André
Augsburg University
Department of Computer Science
Human Centered Multimedia
Universitaetsstr. 6a
86159 Augsburg, Germany
andre@informatik.uni-augsburg.de

ABSTRACT
In this paper, we report on the development of the competitive tangible tabletop game TabletopCars, which combines the virtual world with the physical world. We brought together micro scaled radio controlled cars as active tangibles with an interactive tabletop surface to realize the game. Furthermore, we included Microsoft Kinect[1] depth sensing as an interaction mode for embedded and embodied interaction. Our aim was to investigate the possibilities that emerge through the augmentation capabilities of interactive tabletops for creating novel game concepts and the interaction modes that novel input devices facilitate. This work presents TabletopCars as a testbed for embedded and embodied interaction and describes the system in detail. Finally, we report on a preliminary user study where users controlled the active tangible micro scaled cars through hand gestures.

Author Keywords
Tabletop game, tangible objects, interaction, design.

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Miscellaneous

General Terms
Design, Experimentation, Human Factors.

INTRODUCTION
Small-scale radio controlled cars, also called R/C cars, have always been fascinating and attractive toys for all age groups. R/C cars have in common that they are controlled over a radio-frequency link with a wireless connected controller device. There is a variety of different kinds of R/C cars that depend on the cost, size, power source and engine, achievable speed, or level of detail in terms of resemblance to a real car model. In general, such cars can be electrical powered or nitro/gas powered. Nitro powered R/C cars are meant to be operated outdoors, for example, at large race courses where multiple players compete for the best lap time or to be the first player who crosses the finish line. Actually, the most frequent usage for nitro powered cars are competitions that are organized as professional events by R/C car clubs. Electrical powered models, however, can also be operated within buildings or small rooms since they do not produce exhaust fumes. The sizes of electrical powered models range from scales of 1:5 down to 1:87 or even smaller, whereby the smallest models are called microsizers with lengths around 5cm (see Figure 1). Such models often have limitations, for example, in terms of power capacity or radio control capability. While the bigger sized models have to be operated on large areas, the application of microsizers is limited to small areas due to their weak radio-frequency link. The radio control's coverage of such microsizers is usually up to 10m, which makes them well suited for games that require only little space. An example of such a requirement is driving the microsizers on a table where players are located face to face. This quality is inherently different from the typical car-racing scenario where players usually stand side by side and concentrate on distant R/C cars. The proposed scenario offers chances and qualities known from board games.

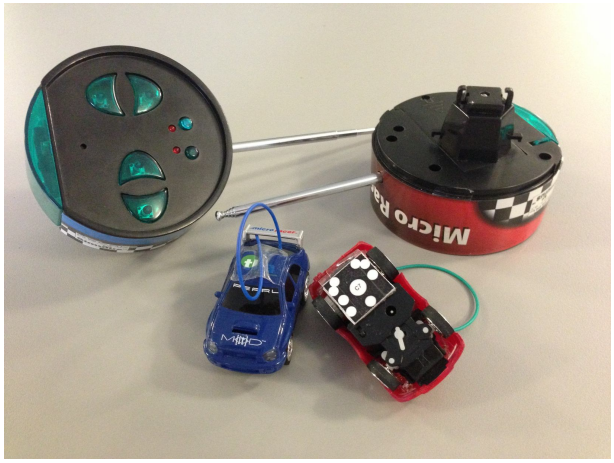

Figure 1. The remote controlled microsizer (right) with their radio remote control and the byte tag on the underside.

[1] http://www.xbox.com/KINECT/ (Sep. '12)

Instead of playing on a static table surface, employing an interactive tabletop such as the Microsoft Surface[2] for games with microsizers offers much potential for dynamic enhancements. In general, interactive tabletops have in common that they display arbitrary digital content on a table-sized surface, which also serves as the input device. The surface of such tabletops is able to sense any kind of objects or contacts on the surface. Thereby, combining microsizers with interactive tabletops provides rich possibilities for creating game designs, novel game concepts, and increased game experiences, for example, through integrating virtual and real world objects into the gameplay as proposed in [5]. Furthermore, this concept benefits from qualities that are known from computer games, such as computer-mediated game management or augmentation of the microsizer depending on their playing context.

This kind of interaction falls into the general category of interaction with tangible user interfaces, for example [6, 4], whereby the usage of tangible objects that are self propulsive in the form of locomotion assigns this interaction into the concrete extension called active tangible interaction [14]. Since users interact with the microsizers locally, the concept is based on *local active tangible interaction* as defined by Mueller et al. [14, p. 171]. Building systems to investigate active tangible interaction is usually attended by laborious handicraft work to fabricate the active tangibles [9] or to establish reliable position sensing of the tangibles such as proposed in the system DMCS [17].

This paper presents an easy and low-cost approach to realize the concept of local active tangible interaction based on microsizers. The proposed design combines microsizers with an interactive tabletop, thus extends the range of applications for microsizers and at the same time offering a testbed for investigations for local active tangible interaction. We implemented an application called TabletopCars that is based on this concept, which consists of four different games for the microsizers. Furthermore, we connected the original radio control units of the active tangible cars to a software layer by utilizing a microcontroller in order to enable embedded and embodied interaction. We describe the whole system in detail and report on issues that arose in engineering such an interactive system. Finally, we present results of a preliminary user study that indicates the general acceptance of embedded and embodied interaction with active tangible microsizers.

RELATED WORK

Games that are based on tangible user interfaces and interactive tabletops have often been the focus for investigations, for example STARS [13], Weathergods [1], PINS [8], IncreTable [10], Optical Chess [20], Comino [11] or Futura [16]. However, only few have been done for active tangible interaction.

On a conceptional level, Jain et al. presented Sketch-a-Move [7] which allowed children to explore the relationship between courses drawn with the finger on small physical cars and the corresponding physical movements of the cars. In comparison with the concept of TabletopCars, Sketch-a-Move involved seperated interaction and execution phases, whereas the interaction in TabletopCars is direct and immediate.

Robert and colleagues [15] built a mixed reality robot gaming platform where the user teleoperated a small robot called Miso. This robot was the active tangible, which moved within a hybrid space that combines physical and virtual spaces by means of displays and projectors. The projection was used to augment the physical space with virtual objects in order to create a mixed reality. Their mixed reality system focused on interaction of a robot with its virtual peers, whereas we address novel interaction possibilities with active tangible microsizers.

Kojima et al. [9] reported on an augmented game environment with small vehicles that is called augmented coliseum where a robot attacked another robot with an augmented laser cannon. Their augmentation of the game environment also included a central part of the tracking system called display-based measurement system, which was the focus of their work.

Tanev et al. presented a system in [18, 19] which connected the radio remote control of a small scaled car to a computer. They tracked the position of the microsizer by means of a live video feed of the gaming environment and developed a driving agent that remotely operated the car through a lap with obstacles. Tanev's work focused on algorithms that enabled the driver agent to optimally achieve the best lap time in a manner that competes with human drivers.

Robot Arena [3] is a system presented by Calife that built on small Lego robots with an interactive table. The position of the robots on the surface was tracked via a top-mounted webcam combined with markers on top of the robots. The presented system represents an infrastructure for the development of novel games and interactive applications. Therefore, the authors concentrated on describing the system design and architecture instead of interaction possibilities and modes.

The work of Haller et al. [5] reported on design recommendations concluded from experiences with several tabletop games. One of the games was called NeonRacer, which is related to TabletopCars in the sense that small cars had to be navigated through a course with tangible obstacles. They showed that everyday objects such as beverage cans or cups can be used to enhance gaming experience on interactive tabletops. In contrast to TabletopCars, the cars in NeonRacer were virtual objects and the interaction was carried out with traditional game pads. In a pilot study, the authors found that players often had orientation difficulties stemming from the car orientation when using the game pad. Many participants would prefer a more intuitive interface for the interaction. Their results motivate TabletopCars, which offers the possibility to connect natural interaction devices such as the Microsoft Kinect depthsensor. Furthermore, supporting embodied interaction also increases player's engagement as

[2]http://msdn.microsoft.com/en-us/library/ee804902.aspx (Sep. '12)

found by in Bianchi-Berthouze [2] or Lindley [12], which promises for novel gaming experiences.

SYSTEM OUTLINE

To realize our active tangible game system, we combined the Microsoft Surface tabletop with micro sized cars from Simulus as sketched in Figure 2. In order to enable embedded and embodied interaction experience, we built a controller unit that establishes wireless control of the micro sized cars through arbitrary input devices and systems, such as the Microsoft Kinect sensor, a standard PC-keyboard, or traditional game pads. The following sections describe each of the components in detail.

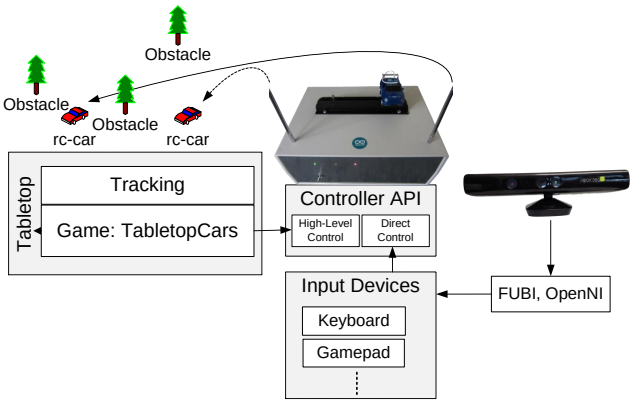

Figure 2. The system outline of TabletopCars.

Microsoft Surface

TabletopCars was implemented for the Microsoft Surface tabletop, which is a commercially available horizontal multi-touch tabletop that provides a development kit for the creation of applications. The form factor conforms to a living room table where people sit around on chairs or couches. The tabletop's vision system offers multi-touch recognition and object tracking based on infrared light technology and rear-projection. The surface measures 24" x 18" with a resolution of 1024 x 768 pixel.

Micro R/C Cars

The micro sized cars (Simulus NC-1195/NC-1196) employed in TabletopCars are miniature models of racecars sized 61mm x 32mm x 28mm (L x W x H). Their engine, magnets for steering control, and lights are electrically powered by a rechargeable battery, which lasts for about 5 minutes of continuous operation and can be charged within 10 minutes. The engine and steering control are radio-controlled over a wireless connection at the frequencies 27 MHz and 40 MHz by means of a proprietary frequency signal scheme. The original radio control unit depicted in Figure 1 provides four buttons for controlling the car. The buttons at the top and bottom produce forward and reverse motion on the back wheels at a constant speed, whereas the buttons at the left and right produce left and right turns on the front wheels. Thereby, these commands afford to drive the car forward or backward at a constant speed while having the choice to drive straight or to turn left or right at a fixed turn radius.

Position and Orientation Sensing

In order to realize a game mangement that maintains a state of the physical world above the tabletop surface, all tangible objects needs to be tracked and identified by the system. Therefore, TabletopCars makes use of the tabletop vision system's capabilities that include detection and tracking of so called byte tags[3] which are also known as surface domino tags. Such tags encode a byte value (0 to 255) and also the direction in which the tag was placed as a geometrical scheme. A byte tag is mounted on the underside of each car in between the rear wheels as depicted in Figure 1. The gap between the tabletop surface and the byte tag is adjusted to about 1mm in order to ensure a stable detection and to prevent friction between the byte tag with the tabletop's surface.

Furthermore, the motor speed of the cars in their original condition was too high for the tabletop's vision system resulting in loosing track of the cars at default driving speed. Therefore, we electrically modified the cars and reduced the motor speed by installing a resistor in between the main power line of the car engine and the battery. This modification reduced the maximum speed of the cars as well as the start-up speed while maintaining full functionality. The resistor has a value of 15Ω which is the best tradeoff between slowing down the cars while ensuring a reliable start-up of the engines. Despite this modification, the cars might still drive too fast for a short time in the rare situation when a car drives over the full distance of the surface without stopping. TabletopCars compensates this rare situation successfully by accounting for short periods ($<= 1sec$) of tracking loss.

Physical Obstacles

In addition to the microsized cars, we employ physical objects as part of the game environment in TabletopCars. The

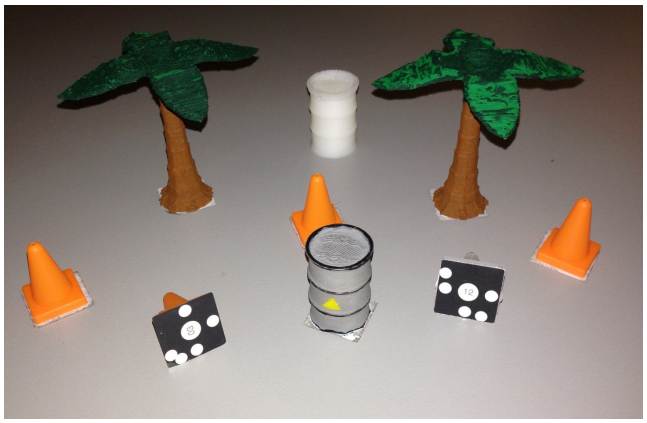

Figure 3. The tangible objects used as obstacles in TabletopCars.

objects act as obstacles which the cars have to drive around, for example miniature models of trees, traffic cones or oil-drums as depicted in Figure 3. These objects are printed out by means of the BFB-3000[4] which is an affordable desktop

[3] http://www.microsoft.com/download/en/details.aspx?id=11029
[4] http://www.bitsfrombytes.com (Sep. '12)

3D printer. The objects are assembled in layers, where the layers consist of either PLA (Polylactic Acid) or ABS (Acrylonitrile Butadiene Styrene) thermoplastic material with a layer thickness of 0.125mm, which also enables printing of detailed and fine grained structures as can be seen for the oildrums in Figure 3. The printed objects are painted afterwards to match the coloration of real objects. Therefore, they are printed with white colored material as white material affords the most neutral background. Finally, the objects are tagged with a byte tag at the bottom side so as to be tracked by the tabletop system.

Another approach to realize tracking is to consider the pattern of the byte tags in the design state, that is to include the pattern on the bottom side of the model, which also works well. However, we decided to use additional byte tags due to more flexibility in case of changing the tag values for particular objects. The proposed process to create physical objects is ideal for realizing tangible objects for tangible interaction, since it enables to model individual objects easily while taking haptics into account. Furthermore, it allows for cost-efficient producing of individual designs that are customized to particular applications.

Remote Controller Unit

In order to build the basis for realizing embedded and embodied interaction, the microsized cars need to be controllable by the tabletop computer. Therefore, we have developed a remote controller unit that connects the original remote controls of the cars to a serial COM-port by means of a microcontroller. The remote controller unit depicted in Figure 4 also houses the antennas, charging indicators, and serves as a charging station for the cars. Furthermore, an external power supply makes the system independent of additional batteries.

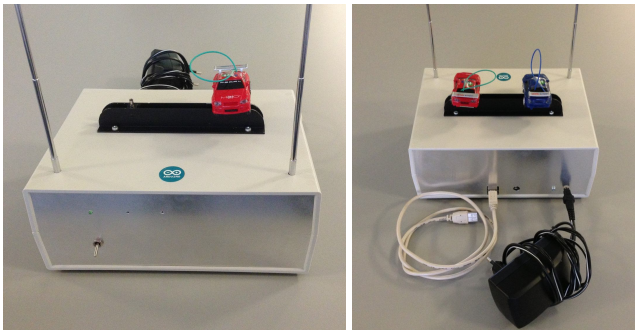

Figure 4. The front- and backview of the remote controller unit.

The remote controller unit is built on the open source rapid-prototyping board Arduino Uno Revision 3[5], which is based on the microcontroller Atmel 328 as the core component. The Arduino platform is widely used for physical computing development due to its simple development environment that employs a variant of the C programming language accompanied by an easy-to-use, rich, and versatile function library. Hence, it enables development of prototypes in quite a short time. The particular microcontroller provides 14 digital in-

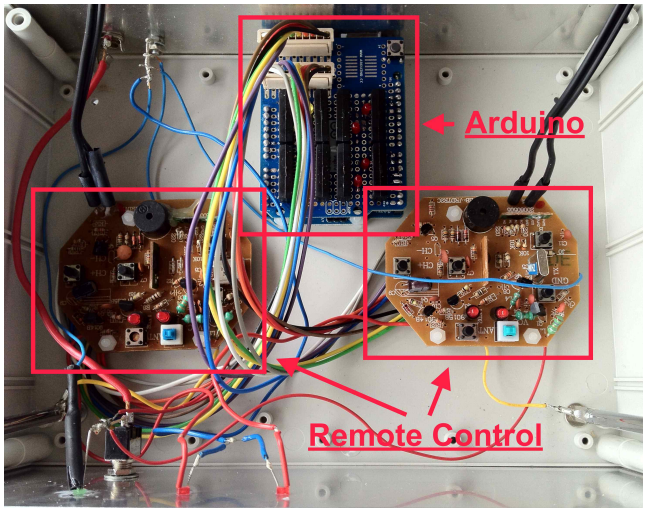

Figure 5. The inner view of the remote controller unit with the microcontroller in the middle and the original remote controls at the left and right.

put/output ports of which eight are used to drive reed-relays that trigger the button presses on the original remote controls (see Figure 5).

Controller API Layers

In order to keep the remote controller unit's operation as stable and reliable as possible, the functionality within the microcontroller's firmware is limited to the basic commands, that is drive forward, drive backward, turn left, and turn right. In addition to that, there is an extended drive command which produces drive pulses similar to a PWM (Pulse Width Modulation) to realize different driving speeds. The remote controller unit provides a USB connection that offers a virtual COM port for communication with a high-level API.

The high-level API is developed as a C#-library that wraps the connection and low-level communication with the microcontroller into a static class. This class enables to submit direct commands to the cars for driving or steering which makes it simple to map input devices such as a keyboard, a WiiMote controller, or a traditional game pad directly to the cars' functions. These functions might also be redirected through a game logic to influence the behavior of a car, for example to match certain game conditions such as imitating the behavior of a car when driving over a sleek or cluttered ground. Furthermore, the library offers higher-level functions such as driving to a given position which involves submitting a sequence of driving and steering commands depending on the currently tracked position for the respective car.

TABLETOPCARS GAMES

TabletopCars is developed as a C#-application employing the Microsoft Surface SDK and provides four different game

[5]http://arduino.cc/en/Main/ArduinoBoardUno (Sep. '12)

modes, which have in common that they have to be played in a competitive manner. Each of the game modes has a time limitation that requires the player to gain as much points as possible within the time frame. In order to start a game, the players' corresponding cars have to be identified by their byte tag value. Hence, each car has to be placed at a given start location as depicted in Figure 6. As soon as TabletopCars has detected the byte tag values and mapped them to the corresponding cars, a timer counts down to zero. The players then have to achieve the goal of the particular game.

The bridge between the virtual and physical world is based on the tracking capabilities of the tabletop system which can be broken in certain situations, such as when a car leaves the tabletop surface, or when obstacles get knocked over by a car. In such cases, TabletopCars pauses the game and indicates the cars' last positions in the virtual world by displaying blinking circles at the particular location on the surface. In order to continue the game, the cars have to be placed at the correct locations after which a countdown starts again.

Car Soccer

Similar to a real soccer game, the Car Soccer game offers a green meadow with big goals at both sides of the play area as depicted in Figure 6. Furthermore, there is a virtual soccer ball that can be moved over the play area. The aim of this game is to achieve points by kicking the ball into the opponent's goal.

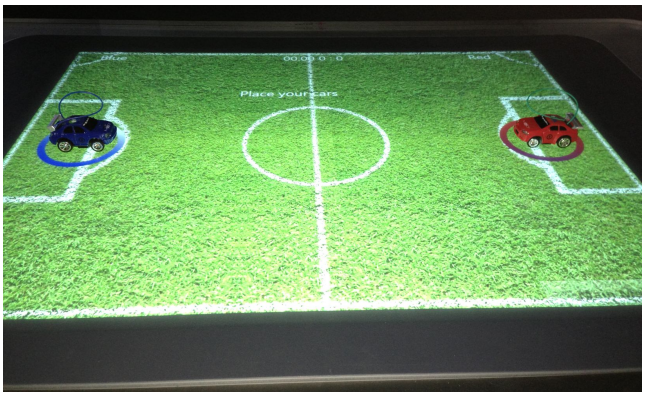

Figure 6. Car Soccer: The initial placement of the cars.

TabletopCars takes the dimensions of the cars into account for a simple collision detection to recognize when a physical car gets in contact with the virtual ball. Depending on the speed of the car and the point of contact, the virtual ball gets either moved or kicked into the appropriate direction, thus gives the virtual ball a realistic behavior that imitates a real soccer ball in terms of physical behavior. Furthermore, a sound for kicking the ball and scoring a goal gives additional feedback to the players.

Car Crashing

Contrary to the Car Soccer game, the Car Crashing game focuses on physical contact of the cars. The aim is to drive with the front of the car into the opponent's car. Points are distributed depending on the parts of the cars where the hit

Figure 7. Car Crashing: The blue car hitting the red car at the side (left); Parcours: An example course with physical obstacles and virtual oil slicks (right).

took place. If both cars hit each other in the front, then both players scores 3 points because none of the players is at an advantage. If one player hits the other at the side (see Figure 7) or at the back of opponent's car, then the player causing the hit scores 5 points because the player is at an advantage and the active part within the crash.

Fastest Lap

One of the most common games with remote controlled cars is driving a car on laps as fast as possible in order to achieve the shortest time for a lap over the opponent's lap times. Fastest Lap implements this game on the tabletop surface where players have to skillfully maneuver their cars as quick as possible through a lap over the track.

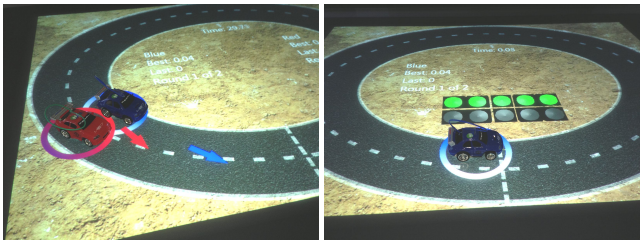

Figure 8. Fastest Lap: Two-player mode (left); Single player mode (right).

The game area consists of a round track on which the cars have to drive, and the start or finish line respectively. Since there is no physical barrier that delimited the round track, the cars can also drive off. Therefore, colored arrows correlating to the colors of the cars assist the players and indicate transit points on the track that needs to be crossed by the cars in order to successfully complete a lap (see Figure 8). For each lap, the lap times are measured and shown together with the best lap time per player. Fastest Lap also offers to choose the amount of laps for each game session within an options menu. The player with the best overall lap time wins the game. Fastest Lap offers a single player mode and a two-player mode as depicted in Figure 8. In contrast to the two-player mode, the single player mode records the actual path that the car drives for each lap and stores the path for the best lap time. Thereby, a virtual ghost car that displays the path of the best lap in the correct chronology accompanies the game as soon as a recorded lap is available.

Parcours

The Parcours game makes use of tangible objects as obstacles (see Figure 3). They are placed on the tabletop surface to compose a course, which the cars have to drive through as exemplarily depicted in Figure 7. Parcours offers two modes of which the first one provides a level editor to create courses with particular obstacle placements, and the second one provides the actual game to play a formerly determined course.

Game rules

In order to successfully pass a course, players have to maneuver their cars through several waypoints as quick as possible and with as few mistakes as possible. The waypoints are connected to a path which is visually indicated by green lines, as shown in Figure 7. Each course has a start and finish line that needs to be passed by the cars whereby they trigger a stopwatch measuring the time taken to pass the course. In addition to the physical objects on the surface, there are virtual oil slicks on the course which the cars have to drive around. In order to determine the winner of a match, the game manages a score for each player, which is calculated as follows. Each passed obstacle scores 10 points and each collision with an obstacle reduces the score by 10 points. In case a car hits an oil slick, the score gets reduced by 5 points. Finally, the score is weighted with the time taken to pass the course in order to determine the overall score.

Level Editor

The level editor enables a player to compose a course by simple placements of tangible objects and touch interaction. The simplest course is defined by a start and finishing line that is also the initial course shown at the start of the level editor. The start and finishing lines can be repositioned and manipulated by touch interaction with the fingers. Each physical obstacle defines a waypoint that needes to be passed by the cars. When placing an obstacle on the surface, the path gets adapted to include the new waypoint and by rotating the obstacle around its z-axis, the waypoint position around the obstacle changes depending on the orientation of the obstacle (see Figure 7). In this phase, oil slicks are added, repositioned, or removed by means of touch interaction.

EARLY USER FEEDBACK

A first version of TabletopCars was presented to eight computer science students in order to gather comments and early user feedback on the design and game concept. This first version had all four games implemented and had to be played with the original remote controls. Extended functionality through the remote controller unit was not taken into account due to early development status. The students learned the usage of the remote control within few minutes and all students preferred the two-player mode due to the higher fun factor when playing against a human player. Overall, the students had never played such a game concept and were really keen on the idea to have active tangible cars enhanced with a digital environment.

All players rated Car Crashing as the most preferred game with the highest fun factor because of the physical crashes of the cars and the act of chasing the opponent's car. The early user feedback showed that users quickly understood the interaction and the game concept with active tangible cars. The user's comments provided us with suggestions for improvements, for example, adapted placement of the point indicators nearby the cars. It turned out that players had difficulties with keeping their score in view when displayed at a static marginal position. Since their attention was focused on the cars, scoring events should be displayed nearby the appropriate cars.

EMBEDDED AND EMBODIED INTERACTION

In order to enable embedded and embodied interaction, the Microsoft Kinect depthsensor was employed as an interaction modality in TabletopCars. The depthsensor captured the whole body of players and tracked a rough skeleton model of them. Using the skeleton model, players can interact with their hand, arm, or body gestures. For TabletopCars, the depthsensor was connected to the game by using the Full Body Interaction Framework[6] (FUBI), which made use of the middleware OpenNI[7] for tracking the skeleton model. FUBI enables easy recognition of static postures or a sequence of postures as dynamic gestures through providing simple XML declarations of the gestures to be recognized.

We realized two simple sets of postures for an initial exploration of such kind of interaction with active tangible cars. The first set makes use of the steering wheel metaphor known from real cars where both hands hold a virtual steering wheel in front of the player as shown in the skeleton model in Figure 9. Turning the steering wheel to the left or to the right is directly mapped to turn commands to the controlled car. Pushing both hands forward or pulling them backward is mapped to forward and backward driving commands of the controlled car. The second set of postures requires only one hand to submit the commands and is based on a virtual "button field" (see Figure 9) spanned vertically in front of the user where the user triggers a button by moving one hand into the area of the button. Depending on whether the player is left handed or right handed, the stop button is adjusted to the left or the right at hip level in order to enable a comfortable initial position of the hands. Each button has a square form with the length of 25cm for a side.

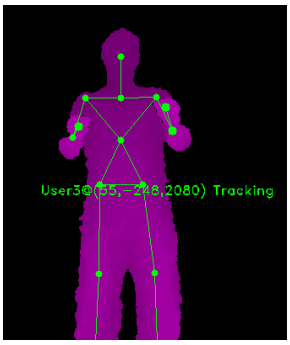

Figure 9. Kinect steering pose (left); Virtual buttonfield (right).

[6] http://www.informatik.uni-augsburg.de/en/chairs/hcm/projects/fubi
[7] http://openni.org (Sep. '12)

User Feedback

For an initial exploration and to get user feedback on embedded and embodied interaction, we conducted a preliminary explorative study to investigate how users cope with this kind of interaction. We compared the presented set of postures with the original remote controls as the baseline in order to get comments on the interaction. Therefore, we recruited 20 volunteers, 17 males and three females, aged from 22 to 34 with a mean age of 28. Each of them was right-handed and all of them were computer science students or researchers.

Procedure

We chose Car Crashing for the exploration because of ratings from early user feedback, which stated Car Crashing as the easiest and most favorite game. Each session was carried out with two volunteers at the same time who played against each other. The players stood side by side at about 40cm distance to each other in front of the tabletop and the Kinect sensor. Both players were tracked by the Kinect sensor and had a clear view on the tabletop. Each volunteer was given an introduction into the game rules and all three interaction modes, that is the original remote control, the Kinect steering wheel metaphor, and the Kinect buttonfield. The order of interaction modes was randomized for each pair of volunteers. After each introduction, the volunteers were given up to 5 minutes to get acquainted with the interaction mode. For this phase, players were instructed to try out each of the commands and to navigate their car for several circular laps over the surface one after another. As soon as each volunteer confirmed acquaintance with the interaction mode, the volunteers were allowed to play one Car Crashing session. Finally, the volunteers were instructed to fill out a questionnaire that asked for comments on and preference of interaction modes.

Results and Discussion

The volunteers were asked to give a preference for an interaction mode where multiple choices were allowed. One of the volunteers preferred only the Kinect steering wheel metaphor and 19 of them preferred the original remote control. Of these 19, three also gave the Kinect steering wheel metaphor as preference and another one would also prefer the buttonfield.

Since players reported in [5] about orientation difficulties with the cars, the questionnaire also asked for the best orientation awareness. The comprehension of the direction in which the car drives is important because left and right turning commands depend on the orientation of the car. Overall, we observed orientation difficulties with all three interaction modes because players corrected their steering commands every once in a while after they realized that the car moved into the wrong direction. Three of the volunteers stated the Kinect steering wheel metaphor as the only preference for orientation awareness, while 17 preferred the original remote control. Of the 17, four also stated the Kinect steering wheel metaphor as preference and two other volunteers also gave the Kinect buttonfield the preference.

The volunteers showed a preference for the original remote control, while 35% stated the Kinect steering wheel metaphor as a preference for the comprehension of orientation. The comments in the questionnaire and in talks afterwards revealed that most players had the feeling of more immediate and direct control with the original remote control compared to the additional latency induced by the Kinect sensing. 50% of the volunteers commented that they would prefer the Kinect steering wheel metaphor if the latency would have been lower because of the novelty of the interaction. The latency from Kinect sensing was not only a technical issue, but also an issue of the amount of movements required to submit a command. Compared to a button press movement on the original remote control, the movement with the hands for the Kinect sensing showed a higher extent, which resulted in the feeling of additional latency. Further comments from players proposed that higher-level interaction techniques would be desired. For example, one hand to point at a position on the surface where the car shall drive to and the other hand for controlling the engine with finger postures.

Overall, all volunteers quickly understood the interaction modes and could cope with them. The preference for the original remote control stems from implementation issues such as latency beside of familiarity with the remote control. Furthermore, the comments showed that Kinect sensing would also be an acceptable interaction mode if the interaction was implemented more intuitively and easier to perform.

CONCLUSION

This paper described the TabletopCars system and its components. We discussed the issues that arose in engineering such an interactive system to enable embedded and embodied interaction. Early user feedback showed that users enjoy the novel concept on games with active tangible cars within a digital environment. The feedback for novel interaction modes that take the whole human hand into account is encouraging in the sense that it would be accepted by users as an input mechanism if the implementation was more intuitive. Such an improvement might be to employ pointing gestures combined with finger postures for the driving commands. In conformance with Haller [5], we also observed orientation difficulties of players when controlling the car, which could be eliminated by using pointing gestures.

FUTURE WORK

Among improvement of the interaction, comments from user feedback during several play sessions suggest to influence the physical behavior of cars under certain circumstances, for example to adapt the behavior of the cars when driving over different virtual grounds or over an oil slick. Therefore, the next steps include redirecting the submitted commands through a game logic to modify the submitted commands in order to adapt the car behaviors so as to adapt to the context in the virtual world. Another purpose of this extension is to enhance game management, for example to stop sending commands to the cars after a physical obstacle has been hit and needs to be repositioned.

ACKNOWLEDGMENTS

The work described in this paper is partially funded by the EU under research grant eCUTE (Reference: 257666). We also thank our students Franziska Mossner, Andreas Seiderer and Matias Schulz for their contribution.

REFERENCES

1. S. Bakker, D. Vorstenbosch, E. van den Hoven, G. Hollemans, and T. Bergman. Weathergods: tangible interaction in a digital tabletop game. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, TEI '07, pages 151–152, New York, NY, USA, 2007. ACM.

2. N. Bianchi-Berthouze, W. W. Kim, and D. Patel. Does body movement engage you more in digital game play? and why? In *Proceedings of the 2nd international conference on Affective Computing and Intelligent Interaction*, ACII '07, pages 102–113, Berlin, Heidelberg, 2007. Springer.

3. D. Calife, J. a. L. Bernardes, Jr., and R. Tori. Robot arena: An augmented reality platform for game development. *Comput. Entertain.*, 7(1):11:1–11:26, Feb. 2009.

4. G. W. Fitzmaurice. *Graspable User Interfaces*. PhD thesis, University of Toronto, 1996.

5. M. Haller, C. Forlines, C. Koeffel, J. Leitner, and C. Shen. Tabletop games: Platforms, experimental games and design recommendations. In *Art and Technology of Entertainment Computing and Communication*, pages 271–297. Springer, 2010.

6. H. Ishii and B. Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '97, pages 234–241, New York, NY, USA, 1997. ACM.

7. A. Jain, L. Klinker, M. Kranz, C. St ”oger, D. Blank, and L. M ”osenlechner. Sketch-A-Move - Design Inspired Technology for Children, 2006.

8. T. Kirton, H. Ogawa, C. Sommerer, and L. Mignonneau. Pins: a prototype model towards thedefinition of surface games. In *Proceedings of the 16th ACM international conference on Multimedia*, MM '08, pages 953–956, New York, NY, USA, 2008. ACM.

9. M. Kojima, M. Sugimoto, A. Nakamura, M. Tomita, M. Inami, and H. Nii. Augmented coliseum: An augmented game environment with small vehicles. In *Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 3–8, Washington, DC, USA, 2006. IEEE Computer Society.

10. J. Leitner, M. Haller, K. Yun, W. Woo, M. Sugimoto, M. Inami, A. D. Cheok, and H. D. Been-Lirn. Physical interfaces for tabletop games. *Comput. Entertain.*, 7:61:1–61:21, January 2010.

11. J. Leitner, C. Kffel, and M. Haller. Bridging the gap between real and virtual objects for tabletop games. *International Journal*, 7(4):1–5, 2009.

12. S. E. Lindley, J. Le Couteur, and N. L. Berthouze. Stirring up experience through movement in game play: effects on engagement and social behaviour. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 511–514, New York, NY, USA, 2008. ACM.

13. C. Magerkurth, M. Memisoglu, T. Engelke, and N. Streitz. Towards the next generation of tabletop gaming experiences. In *Proceedings of Graphics Interface 2004*, GI '04, pages 73–80, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

14. C. Mueller-Tomfelde. *Tabletops - Horizontal Interactive Displays*. Human-Computer Interaction Series. Springer, 2010.

15. D. Robert, R. Wistorrt, J. Gray, and C. Breazeal. Exploring mixed reality robot gaming. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, TEI '11, pages 125–128, New York, NY, USA, 2011. ACM.

16. T. Speelpenning, A. N. Antle, T. Doering, and E. van den Hoven. Exploring how tangible tools enable collaboration in a multi-touch tabletop game. In *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part II*, INTERACT'11, pages 605–621, Berlin, Heidelberg, 2011. Springer.

17. M. Sugimoto, K. Kodama, A. Nakamura, M. Kojima, and M. Inami. A display-based tracking system: Display-based computing for measurement systems. In *Proceedings of the 17th International Conference on Artificial Reality and Telexistence*, ICAT '07, pages 31–38, Washington, DC, USA, 2007. IEEE Computer Society.

18. I. Tanev, M. Joachimczak, and K. Shimohara. Evolution of driving agent, remotely operating a scale model of a car with obstacle avoidance capabilities. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, GECCO '06, pages 1785–1792, New York, NY, USA, 2006. ACM.

19. I. Tanev and K. Shimohara. Towards human competitive driving of scale model of a car. *SIGEVOlution*, 2:14–26, December 2007.

20. A. Wu, D. Joyner, and E. Y.-L. Do. Move, beam, and check! imagineering tangible optical chess on an interactive tabletop display. *Comput. Entertain.*, 8:20:1–20:15, December 2010.