# RoboTable: A Tabletop Framework for Tangible Interaction with Robots in a Mixed Reality

Aleksander Krzywinski
University of Bergen
Fosswinckelsgate 6
5020 Bergen Norway
+47 982 68 173
aleksander.krzywinski@infomedia.uib.no

Haipeng Mi
University of Tokyo
7-3-1 Hongo, Bunkyo-Ku
113-8656 Tokyo Japan
+81-3-5841-6394
mi@itl.t.u-tokyo.ac.jp

Weiqin Chen
University of Bergen
Fosswinckelsgate 6
5007 Bergen Norway
+47 55584143
Weiqin.chen@infomedia.uib.no

Masanori Sugimoto
University of Tokyo
7-3-1 Hongo, Bunkyo-Ku
113-8656 Tokyo Japan
+81- 3-5841-6796
sugi@itl.t.u-tokyo.ac.jp

## ABSTRACT
Combining table top and tangible user interfaces is a relatively new research area. Many problems remain to be solved before users can benefit from tangible interactive table top systems. This paper presents our effort in this direction. RoboTable is an interactive table top system that enables users to naturally and intuitively manipulate robots. The goal of this research is to develop a software framework for human-robot interaction which combines table top, tangible objects, artificial intelligence and physics simulations and demonstrate the framework with game applications.

## Categories and Subject Descriptors
H.5.2 [INFORMATION INTERFACES AND PRESENTATION (e.g., HCI)]: User Interfaces - Input devices and strategies, Interaction styles, Prototyping

## General Terms
Design, Experimentation, Human Factors.

## Keywords
Tabletop, tangible user interface, artificial intelligence, robot, multi-touch, fiducial markers.

## 1. INTRODUCTION
Because the horizontal surface of tables permit the placement of objects, and their large surface area affords the spreading, piling, and organization of these items, digital tabletop user interfaces are becoming increasingly popular in supporting natural and intuitive interactions [8, 15, 17].

A tangible user interface (TUI) is a user interface in which a person interacts with digital information through the physical environment. It provides physical form to digital information and computation, facilitating the direct manipulation of bits [9]. Instead of manipulating virtual GUI elements on the screen through a mouse and keyboard, a TUI invites users to manipulate physical objects that either embody virtual data or act as handles for virtual data. Such physical interactions are very natural and intuitive for humans because they enable two-handed input and provide us with spatial and haptic feedback [18].

The combination of table top and tangible user interfaces is a relatively new research area. Some efforts have been made in this direction, trying to combine table tops with robots and simulations [2, 5, 10, 12, 13, 14], but many problems remain. Existing experimental systems combining table tops with tangible objects are limited in the ability to solve technical problem such as rear-camera tracking of the objects. The main research question of this research is: "how to support tangible interactions on table tops". The ultimate goal of the research is to develop a software framework for natural and intuitive human-robot interaction which combines table top, tangible objects, artificial intelligence and physics simulations and demonstrate the framework with game applications.

An important motivation for this research is the ultimate development of an environment where robot programming can be used in an educational context, allowing users to easily program the robots to perform different tasks, and while having this playful interaction, learn useful concepts that can be transferred to other educational topics. We believe that the table-top interaction will enable us to create an intuitive, yet powerful interface for building the robot AI, and that the subsequent execution of this AI will give instant and valuable feedback to the user, further facilitating the learning experience.

## 2. RELATED WORK
As alternative solutions to the traditional user interface (UI) for human robot interaction, tangible user interfaces (TUIs) have been developed in different forms. TUIs enable users to interact with both physical and virtual worlds [9]. As digital information and functions are coupled to physical objects through TUIs, users are able to manipulate both the physical objects and virtual entities through a natural medium. TUIs provide users an intuitive way to manipulate objects and give an intuitive physical feedback to manipulators as well. Recent researches are trying to develop more intuitive ways for human robot interaction such as finger touch controlling [12] and tangible object controlling [5, 6].

Kato et al. developed a multi-touch interface for controlling multiple robots [12]. In this system, ceiling-mounted cameras are used to track mobile robots on the ground. Users can control the


*Ace 2009*, Oct 29-Oct 31, 2009, Athens, Greece

robot through a multi-touch table, and the robot will directly respond to users' touch command.

Cheng Guo et al. discussed a new method to manipulate robots through TUIs [5]. In this system, users can directly move physical toys on a table to control the corresponding robot remotely. By using two multi-camera tracking systems, all the toys and robots can be tracked so that the robots can respond to the toy movement. However, although these systems provide a remote way for manipulating robots, users will have less chance to interact with the real robots directly since the robot is far away from the manipulators.

In order to give the users not only intuitive controlling experience but also intuitive feedback, there is also some research concerning small robots which can be put on the table surface. Kojima et al. developed a small robot vehicle which can track the projected marker on the table [13]. By projecting some specific marker on the top of the robot, the robot can follow the marker's movement by sensing the marker light intensity. The system also implemented an augment reality game using these small robots on the table.

Based on the marker tracking robot, IncreTable creates an enhanced mixed reality game environment which is developed by Jakob Leitner et al [14]. In this mixed reality system, small robots can influence both physical objects and virtual objects. Besides, the system also provides a mechanism to make virtual events affect the physical world. This system is a good attempt to implement a mixed reality environment on tabletop platform. However, such kind of robot has a limitation that it is not tracked by the system, so it is difficult to move the marker fast since once the robot lose the marker, it will seldom find it again.

More recently, some new projects extend the ability of tabletop human-robot interaction (HRI) applications. For instance, Robot Arena is an augmented reality platform for games which is developed by Daniel Calife et al [3]. This platform allows users to put the robot on a table and interact with it directly. The ceiling-mounted camera can track the color pattern on the top of the robot so that different robots can be recognized separately by different color patterns. This project made a good attempt that united the physical world and digital world and makes real robot interact with virtual robot as well. However, a limitation to this approach is that the patterns used have limited precision and there is an practical upper bound to the number of robots that can be tracked.

The research presented in this paper is inspired by this existing research and tries to combine several of the technologies presented in the related work.

# 3. HARDWARE SETUP

## 3.1 Table

Given our research goal of creating a table-top system that can respond to both tangibles and touch gestures, we found that using a Diffused Illumination (DI) [11] approach could accommodate both design goals, but required extensive calibration to work adequately. In short DI works by illuminating the insides of the table with IR light, and by placing a diffuser on top of the acrylic panel only objects very close to the top surface of the table are illuminated. A schematic view of this concept is shown in Figure 1a). This diffuser will in most cases also double as a projection screen for a rear-projected image. With this scheme fingertips are more difficult to track as they are connected through a finger to the hand, and these objects need to be close to the screen as well, and can cause false positives in the tracking [1].

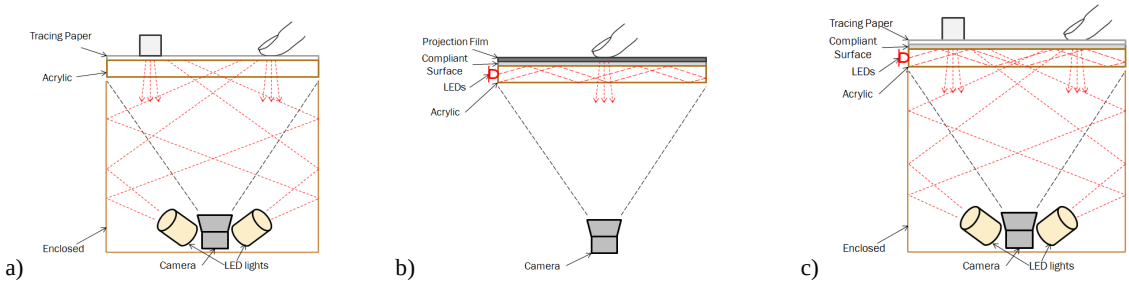


**Figure 1. a) Rear DI setup b) FTIR setup c) Our combined setup**

As shown in Figure 1b), an alternative to using DI for finger tracking is Frustrated Total Internal Reflection [7]. This technique utilize the property of light to be totally reflected inside a medium (in our case an acrylic panel) if it is surrounded by a substance with lower refractive index (usually air), but if an object with a higher index of refraction is placed close to the medium, light will escape at that specific point, and illuminate the frustrating object. This creates a bright spot that a camera can track. Since the FTIR technique only tracks objects causing frustration of the IR-light - requiring a certain pressure - and is not able to recognize any pattern on the tracked object, this method alone was unsuitable for our purposes.

Hence we experimented with a combination of the two techniques, where DI would track markers and FTIR would track finger touches. As shown in Figure 1c), There were certain conflicting requirements of the two techniques that needed to be solved before a successful combination of FTIR and DI could be produced. The main issue was the DI requirement for a very sharp image to precisely read the details of the markers, and the FTIR requirement of having a very bright image so that even very light touches could be recognized.

By experimenting with different aperture sizes, exposure times, frame rates, capture resolutions and fiducial sizes we arrived at a tolerable compromise between required finger pressure and marker recognition. Thus we have a system that both tracks fiducial markers and finger gestures with relative ease, and leverage the strong points of each technique to make a richer, more responsive interaction experience.

## 3.2 Robot

The robots we use on our table are designed with five specific requirements in mind.

1. They must have a small footprint because of the relative small size of the table
2. They need to be highly maneuverable
3. The baseplate has to be as close to the table as possible because of the DI tracking
4. They must be able to communicate with the table
5. They must be aesthetically pleasing and "touchable"

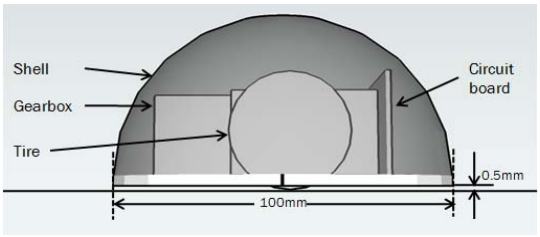


**Figure 2. Mechanical design of robot**

Each robot has a Reactivision fiducial marker carved into the baseplate creating a durable, high contrast symbol that is easily tracked by the camera. The Reactivision fiducials are specifically designed for use on table-top systems, and are optimized for quick recognition and high precision in this environment [1]. There are many other options available, such as the rectangular and highly customizable markers of ARToolKit, but as these are designed to be used in an environment requiring six degrees of freedom, they are not optimized for the special case of table-top use. In a table-top system, the markers are at a practically fixed distance to the camera, and can only be moved with three degrees of freedom.

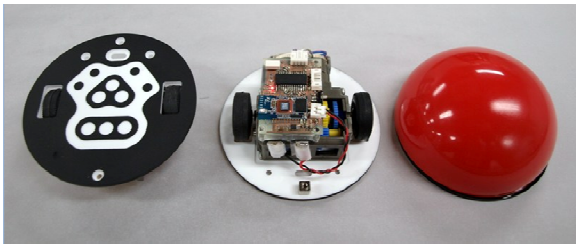

**Figure 3. Robot**
**left: bottom view; center: top view; right: with a shell**

We have chosen Bluetooth technology to connect the robots to the computer in the table, giving us a communication channel that has relatively low power consumption, readily available hardware and software libraries and is able to easily accommodate several units simultaneously. When a robot is put on the surface, the system recognizes the fiducial on the baseplate, and looks up the Bluetooth address of that particular robot in a table and automatically negotiates a connection with that address. The connection process is shown in Figure 4. If the robot is turned on, the connection is seamlessly established within seconds of placing the robot on the surface. The application is thus able to track the position and orientation of the robot on the surface while also giving the robots commands to control its movement.

# 4. SOFTWARE FRAMEWORK

The software framework we have developed for the RoboTable integrates three key technologies in one easy to use package. Very little code is needed to develop simple games from it, and for more complex projects the framework is extendable through public interfaces and simple APIs. It is developed in Java.

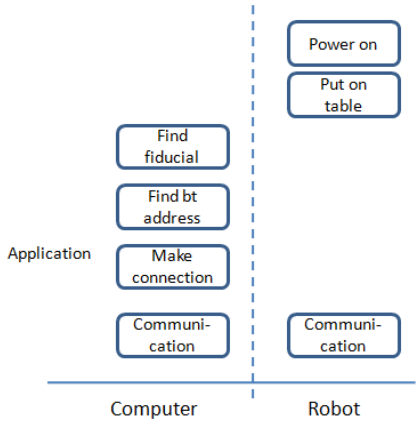


**Figure 4. Robot connection flow chart**

## 4.1 Touch, fiducial and gesture recognition

The ReacTIVision engine handles the segmentation, fiducial, touches recognition and tracking, and delivers a stream of TUIO-data to a specific network address and port. Usually the address is to a local interface, but setups where the workload is divided amongst several computers are easily implemented. Thus, what the TUIO-stream provides are simple add, update and remove events for both cursors (i.e. touches) and objects (i.e. tangibles with fiducial markers). Our system can deliver these data directly to a game application as events if that is desirable, but for touches the framework can also do gesture recognition and provide higher level events.

Sometimes it is desirable that the user draws a symbol or a path on the screen to execute a certain command or provide specific data to the system. For example, the user draws a circle on the surface to access a context sensitive menu at that point. These symbolic gestures can be automatically tracked and recognized by the system, and events will be fired upon completion and successful recognition of a gesture. This is achieved by segmenting the touch path into discrete chunks relative to the path size, and to classify each segment as a movement in one out of eight different directions numbered 0-7, as shown in Figure 5a). Since the notion of up and down, and left and right is dependent upon the position of the user relative to the table, and hence nonsensical to enforce in a table-top system, it must be possible to draw the gesture from any direction. Hence we normalize all the gestures before comparing them to the library of possible gestures assuming that all gestures are drawn “away” from the user as in Figure 5d). The similarities of the gestures are calculated by using the Levenshtein distance between a string generated from the user gesture and all strings stored in a list of applicable gestures as shown in Figure 5e). An event is subsequently sent with the gesture corresponding to the string with the smallest Levenshtein distance, provided that the distance is below a certain threshold. In the future this algorithm may be replaced by hidden markov model-based recognition methods.

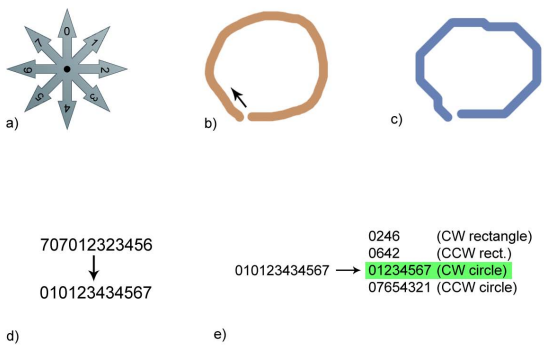


**Figure 5, a) segmentation directions, b) user drawn circle, c) segmented gesture, d) normalized gesture and e) comparison with library**

The system also recognizes gestures involving more than one finger, and reports their relative movement as continuous events. These gestures are triggered when several touch cursors are added in a relatively short time interval, and within a certain threshold distance from each other. We use the approximate diameter of a hand as the maximum distance, but this can be changed to suit the particular applications needs. The values that are reported in the events are the geometric center of the gesture, the positions of each involved touch, the rotation of the gesture relative to the starting position of the touches and the scale of the gesture also relative to the original position of the touch points. Thus the game application can use the derived data directly or use the point data for other means - e.g. painting a polygon shape based on the positions of the fingers.

It is fairly easy to imagine situations where the two techniques for gesture recognition might conflict (e.g. two users start to draw a symbolic gesture each at a location within the threshold distance for a combination gesture. In this case the system will not recognize two symbolic gestures, but one rapidly expanding combination gesture. This could be solved if the system could know which touch came from which user [16, 17] but such recognition is not presently available in our system. Thus we have provided a simple mechanism to subdivide the screen into arbitrary regions (the regions can take any shape) and enable different recognition methods in each region. Granted, this solution does not solve the case where both methods need to be used on the entire surface, but if the interface is naturally segmented (e.g. several images from a photo album scattered on the surface, where each can be independently scaled and rotated - each of them being a region in itself) it is a fair compromise. From a usability perspective, these regions always have to be visualized for the user, or else the interaction will seem arbitrary and frustrating to her.

The algorithms mentioned above have been adopted to also analyze marker input. This, for instance, enables users to make symbolic gestures with a tangible object as well as with a finger. A problem with fiducial markers is that the tracking algorithm has difficulties following very quick movements, and therefore sometimes the tracker believes the fiducial is removed from the table when it is only moving too fast to be reliably tracked. This might result in the marker being reported as added and removed several times in quick succession, thus creating artifacts in the user interface, or impeding the behavior of the application. To alleviate this problem, we have added an algorithm for analyzing the pattern of addition and removal, and finding rapid fluctuations in the status of an object, it will try to interpolate the path of the object instead.

## 4.2 Physical Simulation

To accommodate mixed-reality applications we must ensure that objects on the screen behave in a way that is intuitive for users based on human experience of the real world. Otherwise the users will find it problematic to understand the interaction with the virtual objects on the table, and any advantages in usability leveraged by the familiarity of the user with everyday physics are lost. The flat horizontal surface of a table makes it most sensible to simulate the interactions of bodies on a plane, thus we used a 2D physics simulation framework provided by the Box2D-project. Simulating the extra dimension that would spring up from the surface is superfluous, hence the limitation to two dimensions. The chosen framework has powerful APIs for defining the properties of physical objects, and their interaction with each other when set in motion, thus we can quickly create a world where the objects on the screen collide and exert forces on each other.

As a proof-of-concept application for the physical world we created a simple game that borrows heavily from the classical game of "Pong". Two players use two fingers to create a pad on their side of the table, trying to place this pad where the ball will hit it and bounce back to the opponents' side - hopefully scoring a point in the process. While creating this game we did not have to calculate the effects of the ball colliding with the sides and pads, as all of this was handled by the physical simulation framework. We simply set the physical properties (mass, friction, restitution etc.) of the objects on the table and created the pads where the users touched, while the rest of the game play emerged from the simulation itself.

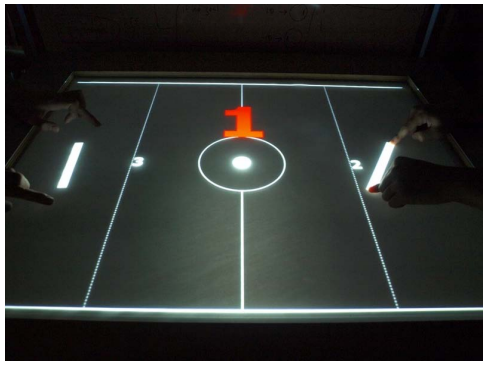


**Figure 6. Pong game**

Granting the real robots on the table a virtual presence is achieved in two steps. Firstly a virtual object with the same dimensions as the robot is created at the position of the fiducial marker and with the same rotation (we use circular robots, but other shapes can be easily be used). When this is done, several "joints" are created between the position of the fiducial marker and the newly created virtual object, and as the marker is moved one end of the joints is moved as well. The joints will then exert forces on the object, much like a rubber band attached to both finger and object, moving it in the virtual simulation and thus forcing it to interact

with the other virtual objects that are on the screen. A proof-of-concept application was created to illustrate the feasibility of this as well, and it takes the form of a similar game to the pong-like game above but where the pads are substituted with tangible objects. The users play the game by moving round, acrylic objects on the table surface, trying to hit the ball and send it at high speeds towards their opponent - not unlike a typical game of air hockey. Thus the manipulation of physical objects has an effect on the virtual world in a very profound way.

The usage of a 2D physics package ensures that there is seamless interaction between the real object and the virtual world. Using a similar method with finger touches is trivial, as the joint is simply created between the position of the finger tip and an already present virtual object. The finger will now exert a force on the virtual object when it is dragged on the table surface.

This enables a quick and easy way to implement the traditional move and rotate-gestures that are the typical show cases of MT systems. The movements of the fingers will rotate or move the virtual object depending on their combined forces and the gesture recognition emerges as an artifact from the physical simulation itself without any specific formulas or algorithms.

## 4.3 Robot intelligence

The robots are capable of a certain level of autonomous action as they contain a behavior arbitration structure similar to, but certainly a simplified version of, the subsumption architecture [2]. The robot has a set of real sensors and actuators, and in addition there is a range of virtual sensors (virtual actuators are also possible, e.g. for controlling a virtual robot arm) that can report data about the virtual world in which the robot is co-situated. These can take the form of radar systems, proximity sensors and table-top positioning systems for accurately reading the position of the robot. Each sensor contains one or more "wires" that can be accessed by the behavior implementations to aid the robot in deciding the correct course of action in the present situation. An actuator has only one wire, which is used to send commands to the actuator.

As a preparation for the future use of a table-top interface to program these robots, we did not want to program the robot AI as classes in Java, as this would be hard to translate to an easy, intuitive interface later on. We rather tried to make some assumptions about the nature of the robot behaviors, and code a simplified programming framework for the robot AI. Since the sensors can do most of the complex calculations required to compute such things as distance, relative speed, predicted positions etc., we decided that the behaviors should consist of one or more if...then...else conditions. When executed the outcome of these conditions decides if the action of the behavior is performed.

Both data values from the sensor wires and constants can be operands in the conditions, while the available operators are equals, not equals, greater than and less than. We envision that future users can easily drag-and-drop the different parts of these conditions into place, thus programming the internal workings of the behavior without worrying about the actual scoping and variable references as these are controlled by the framework.

The actions can be executed in parallel (e.g. send a +10 speed value to both motor actuators - moving the robot forwards) or sequential (e.g. send a -5 speed value to both motors for one second, and then a -10 value to the left motor, and a +10 value to the right motor for two seconds - effectively backing the robot away from an obstacle, and turning it left afterwards to avoid the obstruction).

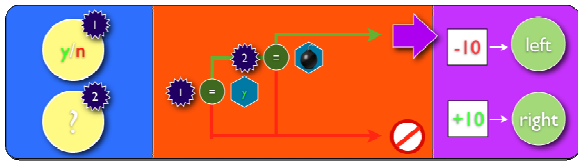


**Figure 7. Behavior for having the robot avoid mines**

Figure 7 shows a simple behavior for avoiding a certain obstacle in the robots path (in this case a mine). The behavior gets its data from a simple radar sensor implementation that has one wire delivering a Boolean value indicating if there are any objects in "beam" (represented by the (1) in Figure 7) and one wire that returns the type of the closest object (represented by the (2) in Figure 7). The logic part of the behavior contains two nested conditionals where the first checks to see if there are any objects in the beam (compares (1) with 'true') and if this evaluation returns true, the second conditional statement is executed checking if the object is a mine (compares (2) with a mine-type). If both these conditions return true, the behavior assumes control and executes the action (in this case turn left), but if one of them returns false, the behavior does not assume control.

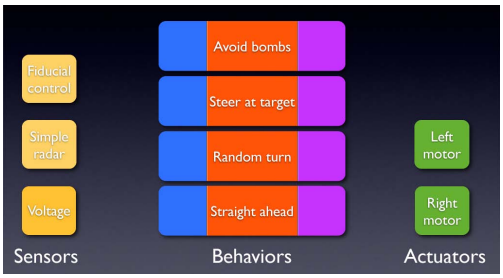


**Figure 8. The behavior stack of a mine-avoiding robot**

Figure 8 shows how a complete AI for a goal-seeking, mine-avoiding robot might be implemented using the scheme outlined above. The topmost behavior (1) is the behavior discussed in the previous paragraph; by scanning the area in front of it the robot looks for mines and avoids them. The next behavior (2) checks if the goal is in the radar beam, and if that is the case, steers straight at it. The third behavior (3) turns the robot in a random direction at random intervals, while the last (4) keeps steering the robot straight ahead. This shows how a rudimentary AI can be given to the robots, and the framework supports future programming of the robots via a table-top interface.

The combination of these three technologies allows us to develop a wide range of applications for our table-top system. In the next section we will look more closely at a prototype application that utilizes all three technologies.

# 5. ROBOBALL

An important milestone in our development of the framework was the construction of a game utilizing all above three key technologies in the game play. The successful implementation of such a game indicates that our integrated framework is suitable for the development of table-top games. We choose to implement a simple soccer game, where two human players control the robots

on the table. Each team presently consists of one robot, but this is due to the availability of robots and not because of constraints in the table framework. The ball is a virtual, simulated object which the robots manipulate by colliding with it, thereby transferring momentum from the real robots, to the virtual ball. This accounts for the mixed-reality part of the game.

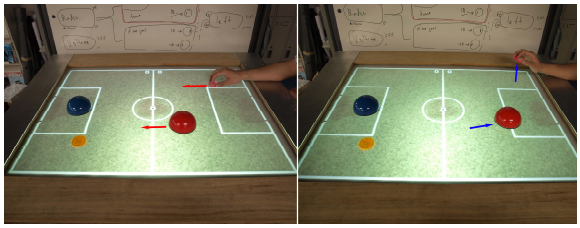

**Figure 9. RoboBall – robot control**

Each robot is controlled by moving a colored acrylic disc on the table surface. When the disc is first placed on the surface, the initial rotation and position is recorded and subsequent movement and rotation of the disc is translated into commands for the robot. Figure 9 shows simple interaction with the robot. The discs are marked with an arrow indicating their main "axis" to the user, enabling the user to intuitively understand which movements will translate to which commands for the robots. Movements parallel to the main axis will increase or decrease the speed of the robot, with movements in "front" of the original point moves the robot forward while movements behind that point will move the robot backwards. Rotation of the disc will result in a difference in the speed of the two wheels, thus steering the robot in the direction of the rotation. If the robot has no forward or backward speed, it will turn in place with a speed proportional to the rotation. This accounts for the tangible input part of the game.

One may argue that this is not the most intuitive way to control a robot on a table-top system. The main argument is that while the control is simple enough for a robot facing the same direction as the user, robots can have an arbitrary rotation compared to the user, thus forcing the user to do a mental mapping between the movements of her hand, and the movements of the robot. For our prototype purposes we argue that this mapping is not a serious usability issue, as a similar problem face people steering radio-controlled vehicles without seriously impairing the swiftness and precision with which these vehicles are controlled.

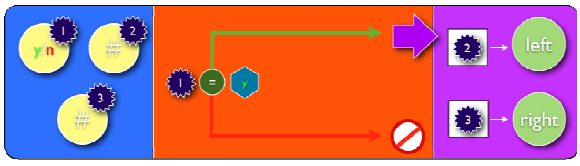


**Figure 10. Behavior for direct control of robot**

The direct control of the robot is provided by a special behavior that resides on the top of the behavior stack. As shown in Figure 10. If a tangible object is present on the table the behavior assumes control of the robot, inhibiting all other behaviors, and executes its action. The action is simply to forward the calculated speed of each motor to the corresponding actuators, whereas the actual calculation lies within a special virtual sensor made specifically for this purpose.

If the tangible object is removed from the table the aforementioned behavior will not inhibit the rest of the stack, and robot's default AI routine will assume control – as shown in the second picture of Figure 9. This routine is implemented using the behavioral stack, and in our prototype game it consists of three simple behaviors. The topmost behavior of the three halts the robot if it is in the proximity of a special marker (invisible to humans) close to its own goal, indicating that the robot is properly on the defense. The next behavior rotates the robot until it faces its own goal, while the last one moves the robot as fast as possible towards the defensive marker. This accounts for the AI part of the game.

In a one robot system, this default behavior is not very useful, but it serves its purpose as a proof of concept of the AI control of the robot. In a two or three robot system, one can see that this simple defensive routine could greatly simplify the game for a human player, as one simply has to relinquish control of one robot to have it quickly move back to a defensive position.

## 6. CONCLUDING REMARKS

In this paper we have briefly described our hardware setup where we have successfully combined the FTIR and DI techniques, leveraging the strong points of each to create an environment where both fiducial markers and finger touch input can be used simultaneously and with a high degree of responsiveness. This setup has formed the basis of our experimentation with a framework for developing applications with tangible user interfaces. In addition to this, we also have included robot control technology in our framework. By developing robots specifically for our experiment based on five criteria, the robots can easily be tracked by the means of a fiducial marker, are small and maneuverable enough for the limited area of a table and has a robust Bluetooth connection to the computer controlling the table. The framework only needs to be supplied with the id of the marker on the robot, and all subsequent communication is handled seamlessly.

The framework is based upon three key technologies, and is an effort to combine these in a package enabling easy development of powerful tangible user interfaces. The first technology is the aforementioned fiducial marker and touch recognition. The framework includes algorithms for gesture tracking and marker movement interpolation, and generally hides most of the inner workings of such systems. The second key technology in the framework includes is a 2D physical simulation, thus enabling developers to, amongst several things, utilize the human knowledge of the real world to help users understand interactions in the virtual world, or use physical simulation as a base premise in their application (as exemplified by our Pong, Air Hockey and RoboBall games). Lastly we have developed a primitive, subsumption based behavioral framework to develop a simple AI to control the actions of the robots.

As a prototype application we have developed a simple game of robot football, where two robots struggle to put a virtual ball in each other's goals. Human control is needed for any proactive action, but if that control is relinquished the robot reverts to the default behavior according to the AI implementation, and assumes a defensive stance in front of its own goal.

Robots have played an important role in education for many years and their presence is stimulating for students [4]. The framework presented in this paper enables us to create an intuitive

and powerful interface that allows users to easily program robots to perform different tasks. Through programming and playful interacting with robots on table tops, students can learn concepts and principles in different disciplines such as mathematics and physics and different levels of education, while also learn to think creatively, reason systematically, and work collaboratively.

This prototype application is a proof of concept that our framework can be used to quickly develop games that combine robots, AI, physical simulation and tangible input, but future development of the framework is needed to fully achieve the potentials of the combination of these technologies.

## 7. FUTURE WORK

The first task we will focus on is the refactoring of the framework codebase. Features have been added to the framework as they were needed in some prototype application, and this almost organic growth has resulted in a framework that contains the features we want, but there is too much coupling of the different components, and some behavior is duplicated and accessible through different parts of the API. We need to refractor the code and provide simpler and cleaner interfaces for application development before we continue our development and evaluation of new and more advanced applications. This refactoring of the framework precedes a future release of the framework under an open-source license.

The RoboBall application will be extended with more robots and an evaluation will be performed to provide some feedback regarding the usability of the tangible user interface. Our work has presently been guided by the assumption that TUIs have high usability - an assumption based on related research - but several user studies will tell us whether this assumption holds true for our specific implementation of a TUI.

If the user studies confirm our assumptions, we will continue with developing the application that is the ultimate goal in our research, a table-top development and testing environment for robot AI that can be used in an educational context to enhance children's learning of important concepts within mathematics and physics.

Creating a TUI to develop and manipulate simple behavior systems is a further possibility within the framework. The goal of such an interface is to have end-users program the entire behavior stack of the robots directly on the table surface itself, utilizing finger touches and tangible objects in the process. This kind of in situ development will hopefully facilitate a playful learning environment where ease of use is combined with immediate feedback of the AI progress at the touch of a finger.

## 8. REFERENCES


[1] Bencina, R., Kaltenbrunner, M. and Jorda S. (2005). Improved Topological Fiducial Tracking in the reacTIVision System. Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops - Volume 03.

[2] Brooks, R.A. (1985). A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation: 2 (1), 14–23.

[3] Calife, D., Bernardes, J. L. and Tori, R. (2009). Robot Arena: An Augmented Reality Platform For Game Development. ACM Computers in Entertainment, Vol.7, No.1, Article 11.

[4] Fong, T., Nourbakhsh, I., and Dautenhahn K. (2003). A survey of socially interactive robots. Robotics and Autonomous Systems, Special issue on Socially Interactive Robots, 42(3), pp. 143-166.

[5] Guo, C., Young, J.E. and Sharlin, E. (2009). Touch and Toys: new techniques for interaction with a remote group of robots. ACM CHI 2009; April 4-9, 2009, 491-500.

[6] Guo, C. and Sharlin, E. (2008). Utilizing Physical Objects and Metaphors for Human Robot Interaction. In Proceedings of Artificial Intelligence and Simulation of Behavior (AISB '08). Aberdeen, England (April 1-4, 2008). AISB Press

[7] Han, J.Y. (2005). Low-cost multi-touch sensing through frustrated total internal reflection. Proceedings of the 18th annual ACM symposium on User interface software and technology (UIST'05). 115-118.

[8] Hornecker, E. (2008). "I don't understand it either, but it is cool" - visitor interactions with a multi-touch table in a museum. In Third IEEE International Workshop on Tabletops and Interactive Surfaces Tabletop 2008 October 1-3, 2008, Amsterdam, The Netherlands. pp. 113-120.

[9] Ishii, H. (2008). Tangible bits: beyond pixels. In Proceedings of the 2nd international Conference on Tangible and Embedded interaction (Bonn, Germany, February 18 - 20, 2008). TEI '08. ACM, New York, NY, pp. xv-xxv.

[10] Izadi, S., Butler, A., Hodges, S., West, D., Hall, M., Buxton, B. and Molloy, M. (2008). Experiences with building a thin form-factor touch and tangible tabletop. In Third IEEE International Workshop on Tabletops and Interactive Surfaces Tabletop 2008 October 1-3, 2008, Amsterdam, The Netherlands. pp. 181-184.

[11] Kaltenbrunner, M. and Bencina, R. (2007). reacTIVision: a computer-vision framework for table-based tangible interaction. Proceedings of the 1st international conference on Tangible and embedded interaction (TEI'07). 69-74.

[12] Kato, J., Sakamoto, D., Inami, M. and Igarashi, T. (2009). Multi-touch Interface for Controlling Multiple Mobile Robots. ACM CHI 2009; April 4-9, 2009, 3443-3448.

[13] Kojima, M., Sugimoto, M., Nakamura A., Tomita, M., Nii H. and Inami, M. (2006) Augmented Coliseum: An Augmented Game Environment with Small Vehicles. IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP' 06).

[14] Leitner, J., Haller, M., Yun, K., Woo, W., Sugimoto, M. and Inami, M. (2008). IncreTable, a mixed reality tabletop game experience. Advances in Computer Entertainment Technology 2008. 9-16.

[15] Mahmud, A., Mubin, O., Octavia, J. R., Shahid, S., Yeo, L., Markopoulos, P., Martens, J-B. and Aliakseyeu, D. (2007). Affective Tabletop Game: A New Gaming Experience for Children. In Second IEEE International Workshop on Horizontal Interactive Human-Computer Systems Tabletop 2007 October 10-12, 2007, Newport, Rhode Island, USA. pp. 44-51.

[16] Morris, M. R., Huang, A., Paepcke, A. and Winograd T. (2006). Cooperative Gestures: Multi-user Gestural Interactions for Co-located Groupware. ACM CHI 2006, April 22-27, 2006. 1201-1210.

[17] Morris, M.R., Piper, A.M., Cassanego, A., Huang, A., Paepcke, A., and Winograd, T. (2006). Mediating Group Dynamics through Tabletop Interface Design. IEEE Computer Graphics and Applications, Sept/Oct 2006, 65-73.

[18] Rosenfeld, D., Zawadzki, M., Sudol, J., and Perlin, K. (2004). Physical objects as bidirectional user interface elements. IEEE Computer Graphics and Applications 2004; 24(1), pp. 44-49.