ToMMI
A Software Library for Multiplatform Tangible Mobile Interaction

Francesco Strada and Andrea Bottino
Department of Control and Computer Engeneering, Politecnico di Torino, Torino, Italy

Keywords: Tangible Interaction, Capacitive Displays, Passive Tokens, Multiplatform Mobile Computing.

Abstract: In this paper we present ToMMI (TOkens for Multiplatform tangible Mobile Interaction), a software library capable of supporting both tangible and multi-touch interaction on mobile devices. ToMMI exploits 3D printed tokens that can be recognized and tracked on devices equipped with commercial capacitive displays. Results shows that the computational core of ToMMI is lightweight and can extract token information in less than a millisecond. Furthermore, the library has an intuitive interface, which facilitates its use, and it is based on a portable framework that guarantees the deployment of any application based on it on a plethora of different smartphones and tablets. We also report that ToMMI code is available, open source, for the research community.

1 INTRODUCTION

The introduction of multi-touch surfaces has rapidly changed the way we access technology and digital information. Nowadays, we use touch to interact with our smartphones, laptops, ATMs and smart TV sets. Multi-touch interfaces (MTIs) are simple and intuitive to use, since they provide a direct and more natural interaction with on-screen contents. Another interesting approach is that of Tangible User Interfaces (TUIs), which enable interacting with digital information through the manipulation of physical objects of the real world (Ishii, 2008).

The integration of these two approaches offers a seamless information representation and interaction that spans both digital and physical worlds and has the potential to improve the user experience, to enhance learning and discovery activities in educational contexts and even to promote collaboration, information sharing and the rise of social experiences (Yu et al., 2011). These opportunities have been investigated in several works, mainly exploiting interactive tabletops. In order to support the implementation process on these devices, several tools and SDKs have been developed and shared with the research community (Jordà et al., 2007; Reactivision, 2016; CCV, 2016). Given the large diffusion of low-cost consumer mobile devices, commercial MTI-TUI applications for these platforms are starting to be available as well (AppMATes, 2016; Tiggly, 2016). However, as opposed to interactive tabletops, there is actually a lack of tools supporting the development in the mobile area. In order to contribute to the problem, this paper describes ToMMI (TOkens for Multiplatform tangible Mobile Interaction), a robust and low-cost solution that facilitate the implementation and multi-platform deployment of MTI-TUI mobile applications.

As we introduced, the problem of using tangibles has been initially tackled in the area of tabletop computing, where tangibles are usually tracked by means of computer vision algorithms that analyze images of a camera placed underneath the interaction surface. To simplify information extraction, tangibles are normally equipped with fiducial markers (Jordà et al., 2007).

With mobile devices, the computer vision approach is clearly unsuitable. However, since these devices are equipped with capacitive screens, one straightforward approach is developing tags capable of emulating finger touches. This can be done either using different conductive materials (*passive tokens*) or circuits (*active tokens*). Passive tokens are usually characterized by a pattern of touch points on their base that allow encoding data like ID, position and orientation (Kratz et al., 2011; Chan et al., 2012). Passive tokens can be easily and cheaply manufactured but they require human contact to be sensed. On the contrary, the fact that active markers are based on different electronic circuits allows them to be sensed without direct user touch but, at the same time, makes them more expensive and technologically complex (Yu et al., 2010; Yu et al., 2011). Moreover, they

Strada F. and Bottino A.
ToMMI - A Software Library for Multiplatform Tangible Mobile Interaction.
DOI: 10.5220/0006141401000107
In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017)*, pages 100-107
ISBN: 978-989-758-229-5

often have a greater dimension compared to passive markers (Voelker et al., 2013), thus occluding larger parts of the display.

As an alternative, approaches based on magnetic sensing have been proposed (Liang et al., 2014a; Liang et al., 2014b; Patten and Ishii, 2007). These tangibles are characterized by a case containing several magnets, whose position is sensed by a Hall sensor grid attached on the back of the display. Hence, a clear disadvantage of this technology is that it requires external hardware to track the tokens.

The approach we propose with ToMMI is based on two elements: (i) the design of custom capacitive passive markers, and (ii) the development of a lightweight software library for identifying and tracking them. Similar to other works, our tokens are characterized by unique patterns of conductive touch points that encode position, orientation and ID. In particular, the token design is the result of a compromise between two conflicting constraints. First, since usual tablets screen require the smallest token to avoid screen occlusion, we tried to minimize their size while guaranteeing a reliable extraction of the required information. Second, since most capacitive screens guarantee a limited number of detectable touches, we pointed at minimizing the number of contact points per token while maximizing the number of unique IDs. This last requirement is also vital to allow an effective integration of MTI and TUI techniques.

The rest of the paper is organized as follows. In Section 2, we will detail the characteristics of the ToMMI tangibles and, in Section 3, of the software library used to manage the MTI and TUI interaction. Finally, in Section 4 we will discuss some real implementations based on ToMMI and, then, we draw the conclusions.

2 TOKEN DESIGN

We start describing the design of the ToMMI tangibles. As we commented in the Introduction, we followed an approach similar to previous works. Our tangibles are capacitive passive tokens capable of conveying the following information:

- an identifier, allowing to discriminate between different tokens;
- the position in screen space of the token;
- the token orientation.

This information is encoded into specific patterns of touch points. The following questions were driving the design of the tokens. First, given that the pattern of touch points of the token should encode the required information in a robust and reliable manner, which can be an appropriate layout that guarantees these characteristics? Then, since current mobile devices limit the number of simultaneous touches to (around) ten, which is the minimal number of touch points that (possibly) enables the contemporary use of multiple tangibles placed on screen and (mandatorily) leaves some touches available for MTI interaction?

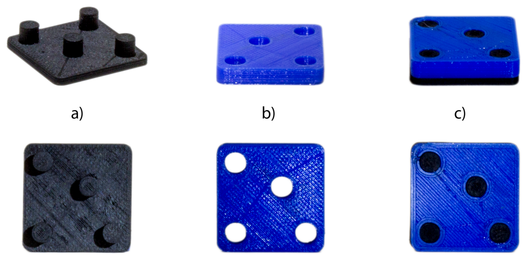


Figure 1: The capacitive tangible (c) consists in a set of contact points (a) enclosed in a PLA shield (b).

Before detailing the design choices taken to answer these questions, for the sake of clarity, we inform the reader that in the following the tangibles will be also referred as *tokens* and their "feet" (i.e. their contact points) as *markers*.

2.1 Prototyping the Tangibles

For the construction of the tangibles we took advantage of the recent availability of conductive graphene filaments on 3D printers (GrapheneLab, 2016), which were used to create the contact points, attached to a common base and enclosed in a plastic PLA shield (Figure 1).

The first analysis was related to define the minimum marker size that allows its recognition on different devices and the minimal distance between two markers that guarantees to recognize them as separate touches. These tests were conducted on devices from different manufacturers, with different characteristics (i.e., both smartphones and tablets) and running different operative systems (iOS and Android).

As for the marker size, we printed several cylinders with a varying diameter in the range $[3,10]$ mm with a step of 1 mm. As a result, we found a marker of 6 mm diameter can be reliably detected on all devices.

In order to identify the minimal distance between two markers, having fixed their size to 6 mm, we created several tokens with two markers, varying their center to center distances in the range $[8,15]$ mm with again a step of 1 mm. Our experiments showed that a distance of 12 mm (i.e., a 6 mm boundary distance between the markers) is sufficient to reliably tell two markers apart on all test devices.

2.2 Marker Types and Token Layout

The solution we propose to address the design requirements introduced at the beginning of Section 2 is using four markers per token.

Three of them (the *reference markers*) define an orthogonal Cartesian reference system capable of providing position and orientation information. The coordinates of these markers into the token reference frame are, respectively, $(0,0)$, $(d,0)$ and $(0,d)$, where d is related to the physical size of the token.

The fourth one, the *data marker*, defines the tangible ID through its quantized position, on a regular grid of 4 mm, in the token reference frame (see Figure 2). However, not all position in this grid are valid. First, the data marker should not be "too close" to the reference markers, i.e. it cannot be placed inside a respect zone of ray 9 mm centered on each reference marker. Second, the data cannot be "too close" to the point of coordinates (d,d), since a point in that position would make the pattern symmetric and, thus, it would be impossible to tell reference from data markers.

As a result, the minimal size that allows for a robust token identification is 30 mm. With this size, the number of unique IDs that can be represented is 8, and a larger set of distinct tokens can be obtained increasing the token size (see Figure 2). The following formulas relate the token size and the number of available IDs. Given an integer number n, a token with size $30+4n$ mm guarantees $8+12n+n^2$ different IDs. For instance, Figure 3 shows examples for $n=0$ (8 IDs) and $n=1$ (21 IDs), while for $n=3$ the token has size 42 mm and it provides 53 unique configurations.

As for the token data, the quantized grid coordinates (i,j) of the data marker define the token ID, the token position is the local $(d/2,d/2)$ point in screen coordinates, and the orientation is given by the angle that the local x axis of the token forms with the global x axis of the screen space. Finally, since recognizing a token requires four touch points, a maximum of two tokens and two finger touches can be recognized simultaneously on a standard tablet.

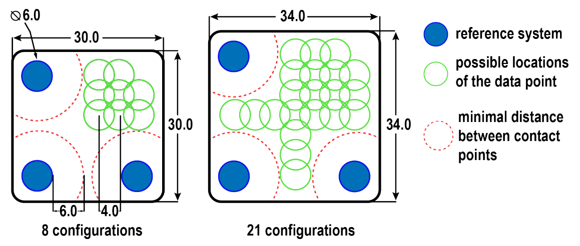

⌀6.0
30.0
30.0
6.0
4.0
8 configurations
34.0
34.0
21 configurations
reference system
possible locations of the data point
minimal distance between contact points

Figure 2: Size of the token elements, and varying number of unique IDs as a function of the token dimension.

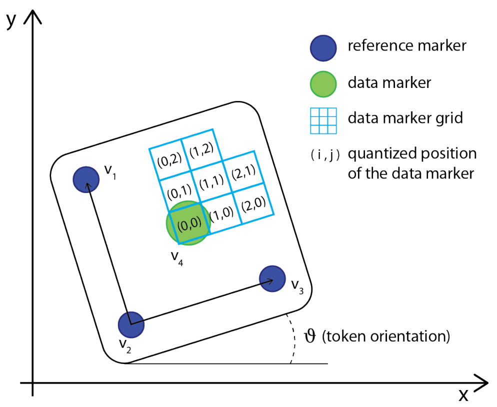


Figure 3: Marker types and properties.

3 EXTRACTING TOKEN AND TOUCH DATA

The software part of ToMMI is responsible of computing all the interaction information related to the fingers and tokens placed on screen. The design and implementation of the library were subjected to the following constraints:

- guarantee the portability on different mobile devices and operative systems;
- provide a simple and straightforward interface, in order to facilitate the integration of ToMMI into the application;
- provide a robust and precise extraction of the information required;
- minimize the computational load.

In the following subsections we will detail these points.

3.1 Portability

In order to tackle this issue, we implemented our framework into Unity 3D, a cross-platform game engine that supports a "write once deploy everywhere" model. This means that an application based on Unity can be developed on any device and then deployed, with minimal effort, into different mobile platforms (Android, iOS, Windows Phone and Tizen). This is possible since the game engine scripting language is built upon Mono (Mono, 2016), an Open Source and cross-platform porting of the Microsoft .NET framework.

Using Unity for the development of mobile apps should not be considered a limitation, since this framework allows the integration of APIs and libraries written in native language and, thus, to access any functionality required by the application.

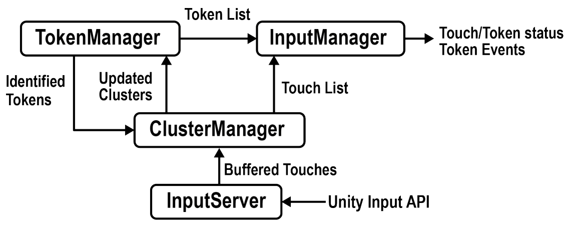


Figure 4: The ToMMI library outline.

3.2 Interface

The interface of ToMMI is pretty simple. The library overrides the basic Input module of Unity, where the main logic of the event system is managed, by introducing an *InputManager* module whose purpose is to store both touches and tokens status and eventually map them into events. Interaction data can then be accessed in two ways.

A first option is to directly refer, at each iteration of the update loop, to the properties of InputManager, which provide:

- the status of all touches during the last frame, which includes position and touch state, i.e. one of the following: *began* (a finger touched the screen), *moved* (a finger moved on the screen), *stationary* (a finger is touching the screen but hasn't moved) and *ended* (a finger was lifted from the screen ending a touch);
- the status of all tokens during the last frame, i.e. an ID and a unique label (which allows using multiple instances of the same token at the same time), position, orientation and token state, where the possible token states are the same as the touch states.

As a second option, the InputManager module defines the following three token events for which any class can define a handler: (i) *OnTokenPlaced*, when a new token is placed on the screen, (ii) *OnTokenRemoved*, when a token is removed from the screen, and (iii) *OnTokenMoved*, when a token is moved or rotated.

3.3 Implementation Details

The design of the ToMMI library is outlined in Figure 4. Since one of the main objective was reducing the computational load, we started by minimizing the number of messages exchanged between the library modules.

Then, since we experienced that the data of the Unity input system might miss some touch ending which, in turns, can result into an incorrect management of the tangible interaction, we introduced an *InputServer* module. Its main purpose is to buffer the filtered touches in the began, updated and ended states and notify the upper layer of any change of these data.

The *ClusterManager* detects and updates touch clusters according to the information offered by the InputServer. The *TokenManager* module identifies new clusters as valid tokens and refreshes the data of tokens corresponding to updated cluster. These pieces of information, together with the list of touches not assigned to a token, are finally handled by the InputManager.

Summarizing, the life cycle of a token is the following. When placed on screen, the ClusterManager create a cluster of its touch point. This cluster is sent to the TokenManager for identification. When the token touch points are moved, the token position and orientation are updated. Finally, when one or more token touch points are marked as ended, the TokenManager flags the token as removed and rises the corresponding event.

3.3.1 Token Identification

In order to be identified as a valid token, the 4 points in a cluster must satisfy the following properties: (i) three of the points $[V_1,V_2,V_3]$, shall define two orthogonal vectors having a common vertex (V_2) and length equal to d, the marker axis length[1], and (ii) the fourth point, V_4, shall be included into the oriented bounding box of the first three points.

Clearly, due to noise in the input data, the verification of these properties considers a suitable margin of tolerance. However, this noise affects as well the accuracy of all token data. While position and orientation errors are less problematic, the noise affecting the data marker can lead to an incorrect identification of the token ID, which is a severe issue.

Thus, in order to reduce the noise and improve the precision, we defined the following linear least square regression problem. Consider the ideal token orthogonal reference system defined by the points $[R_1,R_2,R_3]$, where R_2 is placed in the origin and $||R_1 - R_2|| = ||R_2 - R_3|| = d$. In absence of noise, the two reference systems $[R_1,R_2,R_3]$ and $[V_1,V_2,V_3]$ are related by the following transformation:

$$[V_1,V_2,V_3] = M * [R_1,R_2,R_3]^t \qquad (1)$$

where:

$$M = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & x_t \\ \sin(\theta) & \cos(\theta) & y_t \\ 0 & 0 & 1 \end{pmatrix}$$

[1]Pixel to metric conversion relies on the device DPI

is a roto-translation matrix defined by a rotation θ and a translation (x_t, y_t). Due to noise, M cannot be obtained directly, but it can be computed as the matrix that minimizes the sum of squared residuals between the model (R points) and the observation (V points):

$$min_M \sum_i ||M * R_i^t - V^i||^2 \qquad (2)$$

This problem has the following closed-form solution. For each pair (V_i, R_i) of points, equation (1) can be expanded as:

$$V_{ix} = \cos(\theta) * R_{ix} - \sin(\theta) * R_{iy} + x_t$$
$$V_{iy} = \sin(\theta) * R_{ix} + \cos(\theta) * R_{iy} + y_t$$

and the whole set of equations can be summarized in matrix form as: $A * \overline{x} = \overline{b}$, where:

$$A = \begin{pmatrix} R_{1x} & -R_{1y} & 1 & 0 \\ R_{1y} & R_{1x} & 0 & 1 \\ R_{2x} & -R_{2y} & 1 & 0 \\ R_{2y} & R_{2x} & 0 & 1 \\ R_{3x} & -R_{3y} & 1 & 0 \\ R_{3y} & R_{3x} & 0 & 1 \end{pmatrix}, \overline{x} = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \\ x_t \\ y_t \end{pmatrix}, \overline{b} = \begin{pmatrix} V_{1x} \\ V_{1y} \\ V_{2x} \\ V_{2y} \\ V_{3x} \\ V_{3y} \end{pmatrix}$$

Thus, the resulting solution of the minimization problem in equation (2) is:

$$\overline{x}^* = (A^T * A)^{-1} * (A^T * \overline{b}) \qquad (3)$$

Once we have $\overline{x}^*$, we can update the values of (V_1, V_2, V_3) and, consequently, the marker data (position, orientation and its ID, which is obtained by projecting the data point V_4 on the regular grid defined by (V_1, V_2, V_3). The recognized token receives as well a unique label that, paired with the token ID, allows different instances of the same token class to be used at the same time.

3.4 Computational Load

The entire chain of operations described must be executed within one frame update and its overhead must be minimized in order to provide a lag-free experience to the user.

In order to obtain quantitative results, we performed a series of test where we profiled the framework executing both “normal” and challenging operations, i.e., placing more tokens and fingers at the same time, adding and removing tokens at increasing frequency, performing rapid finger and token movements and so on. Tests were run on both a Nexus9 (2,3 GHz dual-core) and an iPad3 (1 GHz dual-core). The differences between the two devices were minimal if not null. Thus, for the sake of brevity, in the following we report only the numbers obtained with the Nexus9.

Results are summarized in Table 1, which reports the execution time in ms of the various library modules. As for the ClusterManager, we highlight the fact that updating a cluster might involve the creation of novel clusters, a fact that justify its higher computational time. The last two rows of the table show the total execution time required for identifying a novel token placed on screen and for managing a token motion. Both numbers are lower than a ms, which we think justify our previous assertion that ToMMI is a lightweight library.

As final comment, we would like to provide an example of our code optimization efforts. In our preliminary tests, the most computationally intensive process was the solution of the minimization problem in equation 3, with an average execution time of 0.5 ms. To speed up this computation, we first computed the symbolic expression of the result with Maxima and eventually factorized its terms to minimize the number of multiplications required, obtaining a 50x speedup of the initial solution.

Table 1: Execution times.

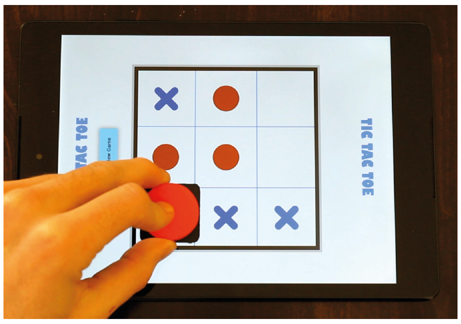
Module	Operation	Time(ms)
InputServer	input update check	0.05
ClusterManager	cluster creation	0.4
	cluster update	0.44
TokenManager	token identification	0.03
	noise reduction	0.01
	token properties	0.05
Token Placed execution time		0.53
Token Moved execution time		0.54

4 RESULTS

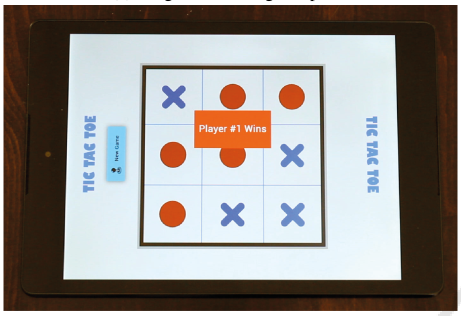
In order to show in more details the capabilities of the proposed approach, in the following we describe the implementation of some applications based on ToMMI.

4.1 Tic-Tac-Toe

The first example is a tangible augmented version of the classic Tic-tac-toe game. Tic-tac-toe is a pencil-and-paper game for two players which take turns marking a 3x3 grid with their own symbol (either a “X”, cross, or a “O”, naught). The player who first succeeds to place three symbols on a row, column or diagonal wins the match. If all marks have been placed without a winner, the game ends with a draw.



(a) Tangible used as game piece



(b) The Tic-tac-toe interface

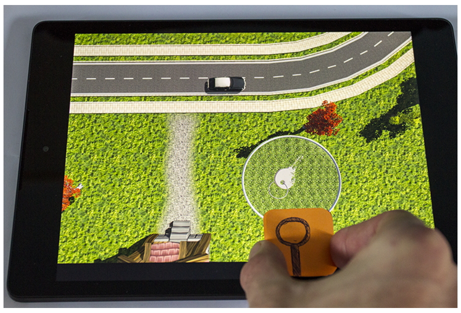
Figure 5: Tic-Tac-Toe.

Given the limit of two tangibles at the same time, players use them as molds to place their mark on the playing grid (Fig. 5(a)). When the game is finished, the players scores are updated and the new game button is enabled to start a new game (Fig. 5(b)).

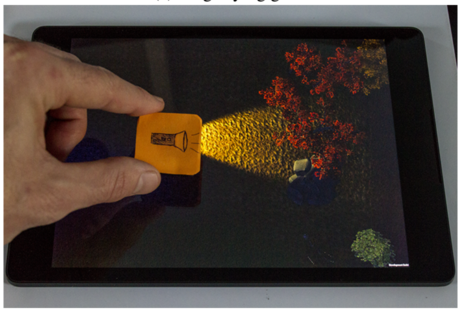
4.2 My Way Home

My Way Home (Bottino et al., 2016) is a game based on interactive storytelling, a form of narrative where the user is not bounded to a linear progressing storyline. On the contrary, he can actively influence the narrative evolution, either by determining the main characters actions or by altering the environment in which the story takes place.

My Way Home is the story of a boy who misses the bus right after school and needs to find a way back home. On his journey, he will face hurdles to overcome and meet characters that can help him to fulfill his objective. The story evolves as a series of scenes. The player can use touches and tangibles to make choices, to control the main character and to exploit tools in achieving her/his goals. As an example, the story features a scene where an old man asks the player to help him find the car keys he lost in the front yard and the player can use a tangible-controlled “magnifying glass” as a helper tool (Fig. 6).



(a) Magnifying glass



(b) Torch

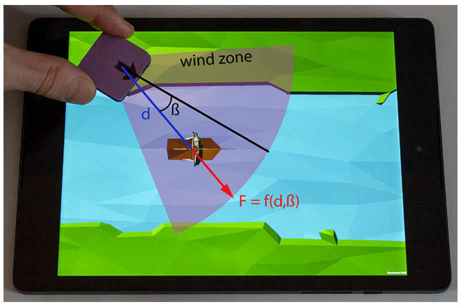
Figure 6: MyWay Home: examples of tangible used as tools to progress in the story.

4.3 Colombo

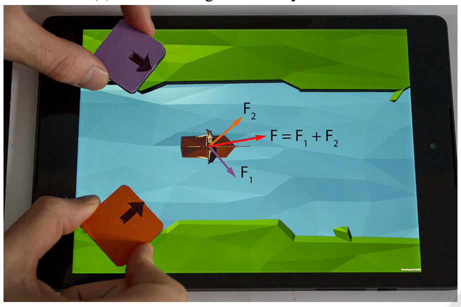
Colombo is an example of a scenario where multiple tokens can be used simultaneously on the same device. Colombo is a collaborative game for two players where the main objective is to navigate a sailboat along a river avoiding collisions with its banks. Each player can contribute to the boat motion with a blower, represented by a tangible playing piece. Direction and intensity of the wind are related to the relative position between the tangible and the boat (Fig. 7(a)). When both player tokens are placed, the resultant of the two corresponding forces is applied to the boat (Fig. 7(b)). Thus, the game requires both communication and coordination between players to fulfill the objectives.

4.4 WaterOn!

WaterOn! (Diniz et al., 2016) is an educational serious game focused on teaching water cycle contents for 8-10 years old children. The game setting is aimed at fostering collaboration and discussion among co-located users. Each player interacts with the game through a tablet, thus being free to move inside the physical game environment. The tablet displays a portion of the whole environment (Fig. 8(b)) and the game features as well a projected screen (Fig. 8(a)) which shows (i) the overall game scenario, where



(a) Wind force generated by a blower



(b) Multiple blowers

Figure 7: Controlling the boat movements in Colombo.

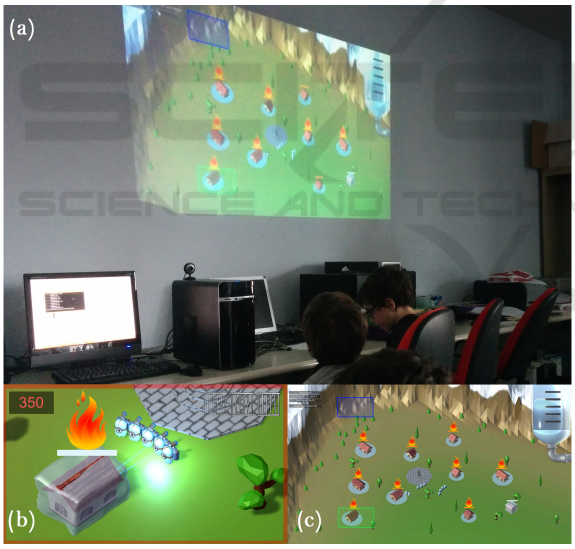


Figure 8: WaterOn!: (a) an image of the projected scenario; (b) view on the tablet screen; (c) feedback of player positions (the coloured boxes) on the main screen.

players are acting as individuals, (ii) the game status, which is aimed at offering a shared understanding of what has been achieved and what has to be completed yet, and (iii) a direct feedback of players' position (Fig. 8(c)).

The game is organized into levels, which introduce the various instructional units defined (i.e., presenting the three states of the water and the transitions between them, describing the movement of water within the water cycle and so on). Both MTI and TUI can be used by the players to perform in-game actions. For example, Fig. 8(b) shows an image of the Melting level, where the villains have frozen village houses and players have to melt them by placing an "heat" token on the house. Multi-touch interaction can be used for both navigating the environment and freezing enemies for a short interval by tapping on them.

Tests with volunteers aged between 8 and 10 showed that all our testers enjoyed the tangibles as interaction tool without reporting neither specific problems in using them nor in combining MTI and TUI.

5 CONCLUSIONS

In this paper we have described ToMMI, an approach for simplifying the implementation and multi-platform deployment of mobile applications that exploit both multi-touch and tangible interaction. ToMMI is based on custom 3D-printed passive capacitive markers and on a software library capable of identifying and tracking them. This library is lightweight, being able to retrieve the required data (ID, position and orientation) in less than a millisecond. It is also characterized by a simple interface, which facilitate its integration, and is portable on different mobile platforms and operative systems. We demonstrated its use through the description of several applications based on it.

Despite the advantages of ToMMI, its main drawback remains the use of passive markers, which require direct user contact in order to be sensed. Thus, as future work, we are planning to investigate the development of (possibly simple) active markers.

We finally underline the fact that ToMMI is an open source software that can be made available to interested researchers upon request.

REFERENCES

AppMATes (2016). Appmates, mobile application toys. http://www.appmatestoys.com. Accessed: 2016-10-06.

Bottino, A., Martina, A., Strada, F., and Toosi, A. (2016). Gaine a portable framework for the development of edutainment applications based on multitouch and tangible interaction. *Entertainment Computing*, 16:53 – 65.

CCV (2016). Community core vision. http://github.com/nuigroup/ccv15. Accessed: 2016-10-06.

Chan, L., Müller, S., Roudaut, A., and Baudisch, P. (2012). Capstones and zebrawidgets: Sensing stacks of building blocks, dials and sliders on capacitive touch screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2189–2192, New York, NY, USA. ACM.

Diniz, A. d. S., Strada, F., and Bottino, A. (2016). Designing collaborative games for children education on sustainable development. In *8th International Conference on Intelligent Technologies for Interactive Entertainment*, INTETAIN 2016.

GrapheneLab (2016). Conductive graphene filament, a conductive 3d printing filament. http://goo.gl/6ZJjqf. Accessed: 2016-10-06.

Ishii, H. (2008). The tangible user interface and its evolution. *Commun. ACM*, 51(6):32–36.

Jordà, S., Geiger, G., Alonso, M., and Kaltenbrunner, M. (2007). The reactable: Exploring the synergy between live music performance and tabletop tangible interfaces. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, TEI '07, pages 139–146, New York, NY, USA. ACM.

Kratz, S., Westermann, T., Rohs, M., and Essl, G. (2011). Capwidgets: Tangile widgets versus multi-touch controls on mobile devices. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, pages 1351–1356, New York, NY, USA. ACM.

Liang, R.-H., Chan, L., Tseng, H.-Y., Kuo, H.-C., Huang, D.-Y., Yang, D.-N., and Chen, B.-Y. (2014a). Gaussbricks: Magnetic building blocks for constructive tangible interactions on portable displays. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '14, pages 587–590, New York, NY, USA. ACM.

Liang, R.-H., Kuo, H.-C., Chan, L., Yang, D.-N., and Chen, B.-Y. (2014b). Gaussstones: Shielded magnetic tangibles for multi-token interactions on portable displays. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 365–372, New York, NY, USA. ACM.

Mono (2016). Mono, cross platform, open source .net framework. http://www.mono-project.com. Accessed: 2016-10-06.

Patten, J. and Ishii, H. (2007). Mechanical constraints as computational constraints in tabletop tangible interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 809–818, New York, NY, USA. ACM.

Reactivision (2016). Reactivision, a toolkit for tangible multi-touch surfaces. http://reactivision.sourceforge.net. Accessed: 2016-10-06.

Tiggly (2016). Tiggly, interactive toys and ipad learning apps for childrens. https://www.tiggly.com. Accessed: 2016-10-06.

Voelker, S., Nakajima, K., Thoresen, C., Itoh, Y., Overgard, K. I., and Borchers, J. (2013). Pucs demo: Detecting transparent, passive untouched capacitive widgets. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '13, pages 325–328, New York, NY, USA. ACM.

Yu, N.-H., Chan, L.-W., Cheng, L.-P., Chen, M. Y., and Hung, Y.-P. (2010). Enabling tangible interaction on capacitive touch panels. In *Adjunct Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 457–458, New York, NY, USA. ACM.

Yu, N.-H., Chan, L.-W., Lau, S. Y., Tsai, S.-S., Hsiao, I.-C., Tsai, D.-J., Hsiao, F.-I., Cheng, L.-P., Chen, M., Huang, P., and Hung, Y.-P. (2011). Tuic: Enabling tangible interaction on capacitive multi-touch displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2995–3004, New York, NY, USA. ACM.