CETA: Open, Affordable and Portable Mixed-reality Environment for Low-cost Tablets

Figure 1: CETA: Mixed-reality system for low cost Android tablets. The tangible blocks represent numbers while joining them represents the addition operation

Sebastián Marichal

Universitat Pompeu Fabra
Barcelona, Spain
sebastian.marichal@upf.edu

Andrea Rosales

Universitat Oberta de Catalunya
Barcelona, Spain
arosalescl@uoc.edu

Gustavo Sansone

Faculty of architecture
Montevideo, Uruguay
gsansone@fadu.edu.uy

Ana Cristina Pires

Faculty of Psychology
University of the Republic
Montevideo, Uruguay
apires@psico.edu.uy

Ewelina Bakala

Faculty of Engineering
University of the Republic
Montevideo, Uruguay
ebakala@fing.edu.uy

Fernando Gonzalez Perilli

Faculty of Information and
Communication
University of the Republic
Montevideo, Uruguay
fernando.gonzalez@fic.edu.uy

Josep Blat

Universitat Pompeu Fabra
Barcelona, Spain
josep.blatt@upf.edu

Abstract

Mixed-reality environments allow to combine tangible interaction with digital feedback, empowering interaction designers to take benefits from both real and virtual worlds. This interaction paradigm is also being applied in classrooms for learning purposes. However, most of the times the devices supporting mixed-reality interaction are neither portable nor affordable, which could be a limitation in the learning context. In this paper we propose CETA, a mixed-reality environment using low-cost Android tablets which tackles portability and costs issues. In addition, CETA is open-source, reproducible and extensible.

Author Keywords

Mixed-reality; open-source; open-hardware; tangible interaction

ACM Classification Keywords

H.5.2 [Information interfaces and presentation]: User interfaces; H.5.1 [Multimedia Information Systems]: Artificial, augmented, and virtual realities

Introduction

Mixed-reality environments allow the combination of tangible interaction and digital feedback, empowering interaction designers to take benefits from both real and virtual worlds. In the learning context, many alternatives for mixed-reality

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

Copyright held by the owner/author(s).
MobileHCI '17, September 4–7, 2017, Vienna, Austria
ACM 978-1-4503-5075-4/17/09.
<https://doi.org/10.1145/3098279.3125435>

Figure 2: Laser cutter making the holder.

environments are being explored, for instance tabletops [10, 4]. However, the portability of these devices is quite limited and the production cost is considerably high. OSMO [2], the mixed-reality play system for iPads, tackles the portability issue but it is a commercial product and is not suitable for low profile tablets, such as the ones distributed by the *One Laptop Per Child* (OLPC) program.

We propose CETA, a mixed-reality environment highly inspired in OSMO [2] that satisfies portability and low-cost requirements. In addition CETA is an open source platform, thus contributing to the digital sovereignty. Thus, the environment (hardware + software) can be adapted for different devices such as smartphones or tablets. This is mainly possible because the full environment is open and reproducible. All the software and hardware (excluding the tablet) are open and available at <https://github.com/smarichal/ceta>. The software is under GPL-3.0 license [1] and the hardware under CC BY-NC-SA 4.0 license [3].

Software Reproducibility and Extensibility Besides the source code is available in a public github repository, extensive documentation is provided to compile and install the system in Android devices. In addition, later in this paper we explain how to extend CETA to use other digital manipulatives and which is the impact in terms of the system architecture.

Hardware reproducibility The hardware design is open and available for 3D printing, laser cut or any other 3D building technique.

Hardware Extensibility It is easily extensible, giving the possibility to customize the tangible objects with moderate programming skills. This means that instead of only using blocks for mathematics learning as we did in [9], it is possible to create alternative designs according to the objectives

of the activity/game (see Figure 12). The steps needed to create new tangibles are explained in the Extending CETA section.

System Description

CETA was developed to promote the use of tangible technologies such as the tablets deployed in public schools as part of the *OLPC* program in Uruguay. The main requirement, and challenge at the same time, was to create an affordable mixed-reality environment for low-cost Android tablets. It was specially designed for a mathematics learning task, thus the tangible elements are inspired in the cuisenaire rods [5], i.e., rectangular blocks ranged in length representing numbers. Cuisenaire rods are widely used for learning basic mathematical skills such as additive composition. In CETA, these blocks become digital manipulatives through augmented reality markers that are detected using the frontal camera of the tablet (see Figure 1).

Hardware

Tablet As explained before, CETA was designed to work in the OLPC tablets. These devices have a Quad Core ARM 1.3GHz CPU, 1 GB of RAM memory and a 0.3 megapixels frontal camera. Their main limitation is the frontal camera, which gives a poor image quality under artificial or low lighting conditions.

The rest of the environment was custom designed and it is composed of:

Holder The holder (see Figure 1) is the wooden structure that maintains the tablet in vertical orientation. It is 10 cm high in order to expand the field of view of the camera enabling a larger interaction zone on the table. The svg file is available for laser cut. It can be easily adapted for tablets with different dimensions although most of the tablets may fit in it.

Figure 3: 3D model of the tangible blocks. Magnets are placed in the small holes to encourage users to join the blocks.

Figure 4: a) 3D model of the mirror b) 3D printed mirror

Mirror The vision of the front camera is redirected to the table using a mirror, and as a result, the tablet can “see” and detect the tangibles on the table. The model of the piece that holds the mirror (see Figure 4) is available for 3D printing and also for laser cut, it might be adapted to the specific dimensions of the tablet (thickness and camera position).

Tangible blocks As mentioned before, the tangible blocks are inspired in cuisenaire [5] rods and have different lengths depending on the represented value, going from 1 to 5 (see Figure 3). We included magnets in the extremities of the blocks in order to provide an affordance to join them, representing the addition concept and the number line representation as discussed in the system’s interaction design [9]. We also provide two versions of the block design, for laser cut and for 3D printers. However, the only requirement for any block to become digital manipulatives is to place the markers on top of it, so we can also build blocks with many other materials and then just paste the markers on top. Thus, the only requirement to create the digital manipulatives is to print the markers and create blocks like the one shown in Figure 8. This makes the system even more affordable and adaptable to different contexts where it may not be possible to access either to 3d printers or laser cutters.

Software

All the software has been developed in Java and it is organized in 3 layers (see figure 7). The third party libraries used are OpenCV [8], libgdx [11] and TopCode [6]. Bellow we discuss each layer and explain the block detection algorithm in layer 2.

Layer 1 - TopCode Computer Vision Module This is the computer vision layer, and in the current design it is implemented by the TopCode library [6]. This is a computer vision library able to identify up to 99 circular markers/tags

(see figure 9). It is specially designed for quick identification and tracking of tangible objects on a flat surface. For each identified marker the library provides: id, location, angular orientation and diameter of the tag. This library has been chosen because it is fast and reliable (works in a variety of lighting conditions), is available for Android, is free and open-source, and recognizes small tags (25x25 pixels). This library has been also used in Strewbies, an Osmo based tangible game for programming learning [7]. Finally, to support alternative detection techniques (based on the shape or the color of the objects, for example) this layer might be modified probably also impacting on layer 2.

Figure 5: CETA supporting partial occlusions. Frames presented in temporal order from left to right. a,b,c are input frames while d,e,f are the detected blocks. Green markers were inferred.

Layer 2 - Detection Module In this layer we implemented the Augmented Rods detection module, this code is specific for the design of our tangible blocks. The detection module

Block	TopCode Markers
1	31, 61, 103, 179, 227, 271, 283, 355, 391, 453
2	93, 117, 185, 203, 793
3	563, 651, 361, 309
4	171, 555, 421
5	1173, 1189, 677

Table 1: Tangible Blocks-TopCode markers mapping. Specific markers must be used for each block.

Figure 6: Top images are the input while the bottom images are the output of the detection module. In frame $t+1$ one of the markers is occluded, so the algorithm computes the candidates $m1$ and $m2$, and after querying the cache determine that the missing marker is $m2$.

and the TopCode library can also be used in desktop platforms, in both cases the input is an image (see Figure 10) that could be loaded from local storage or, as in our case, captured in real time by the camera.

A tangible block is identified as an aligned collection of the same TopCode marker, repeated from 1 to 5 times depending on which block it is. The smallest blocks are those representing the number 1, and they have a single TopCode marker, while the largest contains 5 aligned TopCode markers and they represent the number 5. In addition, we only use a subset of all the available TopCode markers and each marker can only be used within a predefined block, i.e., the marker with id 185 can only be used within blocks of size 2 (see table 1). Next, we use the example in Figure 8 to explain the constraints to be considered when creating augmented rods for CETA:

1. Equal distances between markers within a block, i.e., $d1=d2=d3$. Let's call this distance d .
2. The distance d must be the same in all the blocks.
3. Just one marker id can be used in each block. This is $M1=M2=M3$, let's call this marker M .
4. The marker must be mapped with the value of the block, i.e., in this case M has to be mapped with blocks representing number 3 (see table 1).
5. Background color could be changed. However, the TopCode vision algorithm performs better with higher contrast. The best scenario is black rings and white background.

A tangible object that satisfies the previous conditions is an augmented rod and will be detected by our module implemented in layer 2. Having multiple markers per block is redundant but provides robustness to the block interpretation algorithm. When some markers of the block are not visible, we infer their position using the detected markers and a cache system where we store all the detected markers in the last 5 frames. Using this strategy we are able to support partial occlusions in all the blocks except in the block 1 given that it only contains one marker. The figure 5 shows how the system support partial occlusions. The figure shows three frames in temporal order from left to right, where a,b and c are the input image of the detection module and d,e and f show how the system detects the markers (white), infer the non visible markers (green) and computes the middle point of the block (blue point) and the contour (red rectangles). In Figure 6 we explain the cache algorithm for the specific case of the block 3. Basically, when a marker is not visible the algorithm computes the possible candidates using the distance d , i.e., it computes where the occluded markers could be. In this case, $m1$ and $m2$ are the candidates. Once the candidates are determined, the

Figure 8: Augmented rod design using TopCode markers

Figure 7: CETA software architecture in three layers

cache is queried. The result is that m1 is discarded and m2 is kept as the missing marker of the block 3. Lastly, for each augmented rod the detection module provides: center and vertices (4 points) coordinates, rotation angle, value (from 1 to 5) and dimensions (width and height).

Layer 3 - The Game The top layer contains the code of the game, i.e., an Android application that receives information of the tangibles blocks and updates the state of the game while providing feedback as well. The code in this layer does not know how the blocks are detected. However, given that the input of the detection module in layer 2 is an image, the layer 3 is also in charge of capturing the image from the tablet's camera, for example using the Android API to access the hardware, and provide it to the detection module in the second layer (see figure 10).

Extending CETA

In the mathematics game where we initially used CETA [9], we followed the yellow path shown in Figure 7, i.e., android activity + augmented rod detection module + TopCode library. However, it is possible to extend CETA in many ways. In this section we discuss how extending CETA impacts on

each layer.

Changing the game/activity: In order to use the augmented rods but with another purpose, we just need to make changes in layer 3. This is typically an Android activity which interprets the position and rotation of the augmented rods to update the digital model and provide feedback. In our mathematics learning game [9] we interpreted the rods as numbers and the action of putting them together as the addition operation. It is possible to design other activities where the augmented rods would have other meaning, for example the input control for a game (see Figure 11) or other learning tasks such as magnitude comparison.

Creating new tangibles using markers: It is also possible to design other tangibles using the TopCode markers. In this case the layer 1 remains unchanged but it is necessary to implement or modify the detection module in layer 2 since the layout of the markers might be interpreted in a different way. For example, we could create tangible geometric bodies (see Figure 12) for a geometry learning activity or a module to detect building blocks as it is suggested in the Figure 7. Writing code in this layer is not so complex and offers many opportunities.

Creating new tangibles without markers: The most difficult but also the most powerful extension is to change the computer vision algorithm in order to detect the tangibles using a different approach. This would enable, for example, to detect objects by their shape or color putting aside the markers. This implies a change in layer one (computer vision algorithm) and is also very likely to impact on layer 2 since the output of layer 1 will probably be different.

Figure 9: TopCode tags

Figure 11:
Possible use of the augmented rods to control a game.

Figure 12: Hypothetical tangibles design for geometric bodies using TopCode markers.

Conclusions and Future work

We proposed a low-cost and portable mixed-reality environment. Both hardware and software are open and reproducible. We expect that these efforts mean a step forward in the inclusion of tangible interaction in classrooms in order to take advantage of the technology deployed worldwide by programs such as *OLPC*.

As future work we expect to build high level tools allowing people without programming skills to design tangibles (such as teachers in many cases) and create customized educational apps. We also encourage to the community to build their own projects using CETA and to improve it.

Figure 10: Information flow: The captured frame is scanned by the TopCode library which detects the markers that are interpreted by the augmented rods detection module to compute the augmented rods. Lastly, the game updates its internal state and provides feedback.

ACKNOWLEDGMENTS

This project is funded by the National Agency for Research and Innovation of Uruguay through its Digital Inclusion Grant (Fondo Sectorial "Inclusión Digital: Educación con Nuevos Horizontes" - 2015: FSED_2_2015_1_120888)

REFERENCES

2007. GNU General Public License. (2007). <http://www.gnu.org/licenses/gpl.html>
2013. Osmo. (2013). <https://www.playosmo.com> Accessed: 2017-05-15.
2013. ttribution-NonCommercial-ShareAlike 4.0 International. (2013). <https://creativecommons.org/licenses/by-nc-sa/4.0/>

4. Alissa N. Antle, Alyssa F. Wise, and Kristine Nielsen. 2011. Towards Utopia: Designing Tangibles for Learning. In *Proceedings of the 10th International Conference on Interaction Design and Children (IDC '11)*. 11–20.
5. Georges Cuisenaire. 1968. *Les nombres en couleurs*. Calozet.
6. M Horn. 2012. TopCode: Tangible Object Placement Codes. Retrieved from <http://users.eecs.northwestern.edu/~mmhorn/topcodes> (2012).
7. Felix Hu, Ariel Zekelman, Michael Horn, and Frances Judd. 2015. Strawbies: Explorations in Tangible Programming. In *Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15)*. 410–413.
8. Itseez. 2015. Open Source Computer Vision Library. <https://github.com/itseez/opencv>. (2015).
9. Sebastián Marichal, Andrea Rosales, Fernando Gonzalez Perilli, Ana Cristina Pires, Ewelina Bakala, Gustavo Sansone, and Josep Blat. 2017. CETA: Designing mixed-reality tangible interaction to enhance mathematical learning. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct (MobileHCI '17)*.
10. Nesra Yannier, Kenneth R Koedinger, and Scott E Hudson. 2013. Tangible collaborative learning with a mixed-reality game: Earthshake. In *International Conference on Artificial Intelligence in Education*. Springer, 131–140.
11. Mario Zechner and others. 2013. Libgdx. *Computer software. LibGDX. Vers 1, 1* (2013).