A SOFTWARE FRAMEWORK FOR THE DEVELOPMENT OF PROJECTION-BASED AUGMENTED REALITY SYSTEMS

Marc Sunet, Marc Comino
ViRVIG Group – UPC
C/ Jordi Girona, 1-3, Barcelona

Dimosthenis Karatzas
Centre Visió per Computador
Edifici O, 08193, Bellaterra

Antoni Chica, Pere-Pau Vázquez
ViRVIG Group – UPC
C/ Jordi Girona, 1-3, Barcelona

ABSTRACT

Despite the large amount of methods and applications of augmented reality, there is little homogenization on the software platforms that support them. An exception may be the low level control software that is provided by some high profile vendors such as Qualcomm and Metaio. However, these provide fine grain modules for e.g. element tracking. We are more concerned on the application framework, that includes the control of the devices working together for the development of the AR experience. In this paper we present a software framework that can be used for the development of AR applications based on camera-projector pairs, that is suitable for both fixed, and nomadic setups.

KEYWORDS

Augmented Reality, Modular design, Software framework.

1. INTRODUCTION

Augmented reality (AR) is a growing field that entered mainstream mainly thanks to the ubiquity of mobile devices. However, relying on the use of a mobile device for AR puts some limitations on the features that such a system may provide. Projection-based augmented reality, on the other hand, permits setups that let the user free hands to perform all sorts of interactions, such as virtually writing on physical documents. We are concerned on the development of systems for Human-Computer Interaction (HCI), and we will show examples of applications that provide different types of interaction using this approach. Among the different possible projection-based AR systems, we are more interested in fixed or nomadic setups. A key advantage is that since fixed setups do not require that the user moves cameras or projectors, such systems are easier to calibrate and less prone to accidents. Nomadic solutions [Huber et al. 2012] are in between fixed systems and mobile projected user interfaces [Huber, 2014]. They use pico-projectors that are placed on a fixed position for the duration of the experience. Therefore, they share the advantage of fixed systems in terms of usage: once set up, the user may have her hands free, which provides more flexibility for the interaction. On the other hand, like with mobile projected user interfaces, these are harder to calibrate. We will demonstrate our framework on a nomadic and a fixed setup.

Most setups use special purpose libraries and programs that have been developed hardware-dependent. This hinders the reproduction of an equivalent system in a different place if any of the hardware components are changed (e.g. substituting a projector because larger resolution or brightness is required).

To address this problem, we have developed a device independent, modular software framework, that abstracts the hardware layers into modules, and facilitates the substitution of the any module (camera, projector, input device) with little effort. The system also abstracts the capture and visualization modules. This way, the input can be addressed by naked hand gestures, or with other input devices, and the output can be carried out by simply drawing images or text, or with a more complex set of widgets able to simulate a full-featured virtual desktop. The key modules of our system are: *Hardware Abstraction Layers, Data Abstraction Layer, Communication Protocol, Visualization Module, Interaction Module, and Application Logic*. These components are sufficient to implement a vast amount of different setups, and most of the configurations can be achieved with little changes. Some applications will require extra modules, as we will see later when we

describe some application examples. In the following, we will describe the different parts of the system, the two different setups we built based on this framework, and demonstrate its utility using an augmented document demo application to play music.

2. OVERVIEW OF THE SYSTEM AND RELATED WORK

The field of augmented reality is continuously evolving. With the explosion of mobile devices, augmented reality has gone mainstream, with users of all backgrounds using it for a wide variety of uses such as maps navigation, museum guides, and a bunch of professional applications. Most systems are commonly implemented as see-through systems. This imposes the limitation of requiring a device to be placed between the user and the reality, and sometimes its manipulation is cumbersome or poses limitations on the user freedom. On the other hand, projector-based augmented reality, does not let the user freely change its location, but it may essentially free her hands so that a wider set of interactions may be available (e.g. the Sprout PC [Hewlett-Packard, 2015] by HP). Despite the great variety of such systems, software is far from standardized. Many examples are proprietary, and others are just research-based demonstrations, with the focus placed on the interaction or visualization features, more than the software architecture that makes them possible.

Other previous research has focused on similar problems with a lower degree of generalization, such as in the case of the CAMPAR framework [Sielhorst et al., 2006] tailored to the operating room, with a special emphasis on the synchronization of devices. The approach by Kolomenski [Kolomenski 2013] is similar to ours in the devices used (camera, projector, IR pens…), like other systems [Linder and Maes, 2010, Mistry and Maes, 2009, Weiley and Adcock, 2013], but here we concentrate on the software modularization part. We do not focus on robot-operated systems (i. e. [Tsuji et al., 2013, Bernier et al., 2012]), since our approach is intended to be closer to a nomadic system. We also focus on projected-based AR instead of see-through approaches [Spindler et al., 2012], or systems that require external worn devices [Kim, 2012], since the environments we are interested on (e. g. public libraries), require freedom and little number of external devices. Freehand interaction promotes experimentation, and facilitates user rotation. Moreover, the lack of mobile parts improves the durability of the setups.

Our system consists on a set of decentralized modules that communicate to each other with the use of a communications system (see Figure 1-right). In this system, several channels are open, and the modules can freely register to receive the messages of the different kinds of information.

- **Hardware abstraction layers:** A set of modules are used to hide the nitty gritty details of the hardware specific components from the rest of the system. They thus allow the substitution of a camera or projector element without affecting the rest of the system (Figure 1-left, bottom modules).
- **Communication protocol:** Passing message system that the different modules connect to. It allows is easily parallelization, with different modules on different platforms (computers, mobiles, servers) in a transparent way. Figure 1-right shows and example of massage passing.
- **Interaction module:** The interaction module tracks the user input and issues messages corresponding to the different interactions that are detected (Figure 1-left, second row).
- **Visualization module:** This system is in charge of the rendering of the different elements to be visualized (Figure 1-left, second row).
- **Data abstraction layer:** It is in charge of the input and output of the data that has to be red/written from/to disk (see Figure 1-left, top left module). In many cases this module will be a simple one, but in some others, it might imply working against a more complex database system.
- **Application logic:** This component is the one that defines the current running application. Again, the communication with the other components is handled via messaging (Figure 1-left, top right module).

Apart from the fixed modules, which are common for most applications, other, extra modules can be implemented. Most of these will be application-specific, and we will not deal with them in this paper. We only mention them here for completeness, and they may appear in some examples later.

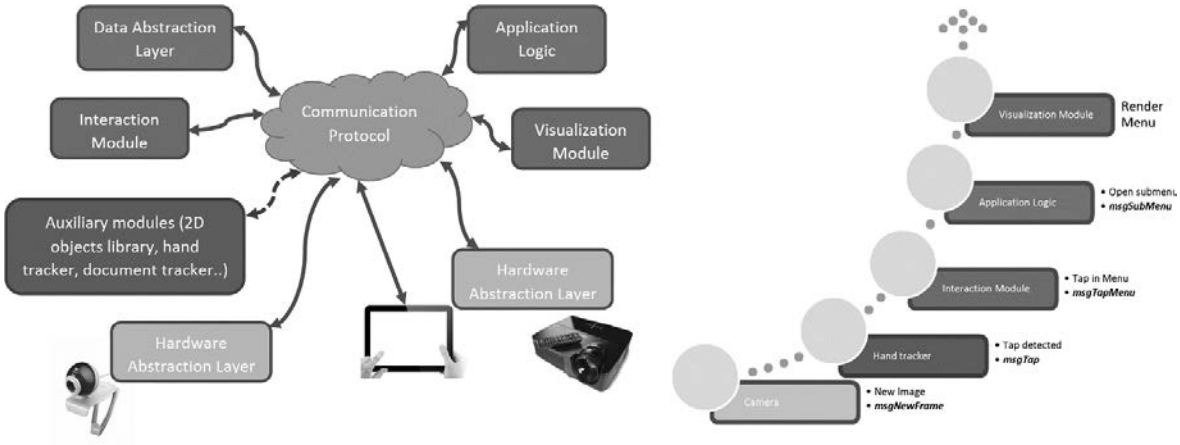

Figure 1. Architecture of our framework. The architecture of modules is shown in left scheme. Right image shows a full example of message passing throughout the system. Upon a hand gesture by the user, a tap, the system generates a submenu. The different modules issue messages into the system and the application logic decides the actions (e.g. opening a submenu) also through message passing.

3.1 Developed modules

For the realization of our system, we developed the following modules: Projector HAL, Camera HAL, Application Logic, Communications Module, Interaction Module, Visualization Module, and the Data Abstraction Layer.

The **Hardware Abstraction Layers** are pretty simple, they abstract the access to images in the case of the camera, and the projection in the case of the projector. The camera images are queued in a buffer, and the interested modules can read them. The **Application Logic** is different in each case. However, since it makes strong use of the other modules, it commonly requires few lines of code. The **Communications Module** is the skeleton that vertebrates the whole system. All the information that is captured or generated is put into the communication system, and the modules that require it, register to the convenient channels. It has been implemented using Google Protobuffers [Google Developers, 2015] over a ZeroMQ [iMatrix Corporation 2012] transport protocol. Protocol Buffers are a language and platform-neutral system for serializing structured data. They are also extensible, which makes them quite useful in many communication systems. ZeroMQ is a messaging transport protocol is a transport layer protocol for exchanging messages between two peers over a connected transport layer such as TCP. In our system, all the modules that may generate data or commands puts messages into a channel, and the modules that wish to read this information only have to register to those channels. This way we achieve a hardly hierarchical structure that is easy to maintain and whose modules can be replaced simply.

We can see a working example of this protocol in a subset of the modules in Figure 1-right. In this example we show how a tap in a menu can be interpreted by our system and generate a submenu. The top-left image is a clipped version of the whole system, where only the modules intervening in the tap processing and reaction are shown. The bottom right part, encodes the different modules with the same colors to facilitate the reading. First, the camera generates a new frame where the hand is tapping on a concrete region of the working space. The camera HAL, as expected, generates a message with the image as the contents. The hand tracking auxiliary module reads the image and detects a tap. This tap is then passed to the system through another message (*msgTap*). The interaction module receives this message and passes the information to the application logic, which is aware of the elements that have been rendered (this can be achieved directly or through previous check with the visualization module). Then, the application logic decides that a new submenu must be opened, and passes this information to the visualization module as a new message that carries out a command, *msgSubMenu*.

For the sake of the reader, we have avoided a thorough description on the parameters and the current format that each message may carry, but there are easy to imagine.

The **Interaction Module** is the one in charge of getting the input from the user and convert this input into commands or information that is broadcast to the interested modules. We have implemented it in two different flavors: hand-based gestures, and IR-pen gestures. In Section 5 we provide more details on the interaction modules. The **Visualization Module** renders all the objects that are projected onto the working area. We have created a lightweight UI library built on top of SFML graphics tool [Gomila, 2015]. SFML is a multi-framework library that provides a simple interface to various multimedia components of the operative system. The principal feature of the UI we have developed is that it allows the 3D rendering of 2D widgets in order to correct the projection deformation induced by an arbitrarily-tilted projector. Moreover, it also serves as a pipe between the gesture module and the application, namely it detects on which widgets the gestures are performed and forwards this information. Finally, an image-based positioning algorithm has been implemented in order to avoid widgets from leaving the workspace when the object from which they are hanging is moved. Adjusting a resolution parameter finer positioning can be achieved at the expense of an increased processing time.

The *backoffice* system, **Data Abstraction Layer**, deals with persistent data. In one of the use cases we developed, for example, we dealt with documentary information. As a result, a database was required, in this case we used an Oracle database of documents with hand generated annotations. The result of the interaction with the application also generated a set of new annotations. These were also stored along the database. This required a module to handle this data. All of this can be abstracted from the application, and in some particular cases, where the data lacks the generalization of the framework we propose, may require slightly more effort, but most common data will be treated simply by a generic data abstraction module.

These developed modules are common to all applications and only little modifications to to some of the systems may be required if we change the input or output devices. In our case, we did not have to change anything for the transition of our nomadic system to the fixed system.

Each application will use all of the previous modules, but the *Application Logic* is dependent on the application to be developed. Therefore, it will be different for every application, but the other components can be simply used as is. Together with these modules, we found that other components can be commonly required in many scenarios, these are enumerated here:

- **Rendering subsystem:** For the visualization part, several strategies can be used, in our case, we developed a library of visual objects and a library of visual feedback elements.
- **Document tracker:** When the application scenario is intended to simulate a virtual desk, the tracking of documents becomes a must. Therefore, this module may be of great utility.

Some of the scenarios we worked with throughout the development of the project dealt with documents. In some cases, the scenario consisted in augmenting the document, by adding some information on demand, and in some other cases, the document was used as input (for identifying or capturing images, etc.). In all these cases, apart from the concrete software for capturing or identifying elements, there is the need of tracking the document in the scene. Therefore, a simple document tracker was implemented and used to provide information both for the input (e.g. capturing information) and output (e.g. projecting extended information onto the document) systems.

3.2 Interaction

The interaction with our system can be carried out using two different techniques: hand-based, and with IR-pens. The most important advantage of the hand-based interaction is the lack of external elements. However, the most important limitation, is the hand segmentation. Since each user may have a different skin color, and the illumination conditions change along the day, the hand interaction lacks some degree of robustness. This is especially true, and may be a problem, for nomadic systems. Unfortunately, since in most places, we are not able to control the illumination totally, this may become a problem, although not as severe, for fixed systems. Ideal conditions should ensure the light is constant along the day, which is not common in most places.

On the contrary, the IR-pens require a third camera with its extra calibration stage. This is an extra element (though still maintaining a low cost at the hardware part). However, they present less problems when interacting because with the IR signal, the system is less dependent on illumination condition changes. Moreover, the calibration stage is quite simple since it does not suffer the illumination changes and therefore is quite straightforward.

In any case, our system has the gestures implemented in the interaction module, and they are independent on the way they are captured. That is, the same gesture can be performed by a hand or by an IR-Pen, in our

case, since these two trackers were implemented, but it would be easy to perform equivalent gestures with other external devices such as the MYO Armband or the Leap Motion, the only issue is the concrete gesture tracker, but the interaction module remains the same.

In order to properly determine the gestures, and to maintain uniformity, the gestures are performed in three stages, as shown in Figure 2:

Initialization: The gesture is detected and identified. Initial visualization cues are provided.

Updating: Gesture is performed by the user. Visual cues identify and communicate the gesture to the user.

Finish: Gesture finishes. If an action is linked to the gesture, it is triggered.

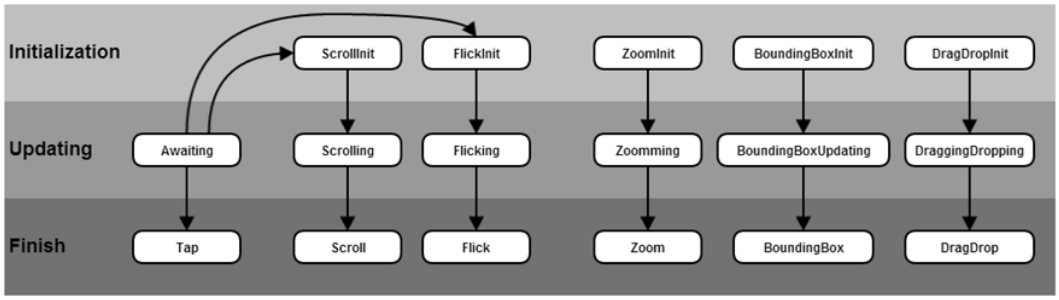

Figure 2. The different stages of the gestures that can be determined by our system.

Note that the figure determines a state machine that is updated throughout the gesture tracking. Each of the boxes correspond to messages issued by gesture tracker to the system. Therefore, the interested modules can read them and act accordingly. In our case, the Visualization Module is aware of the gestures being carried out and generates the appropriate visual cues to inform the user that a certain gesture is being detected and where.

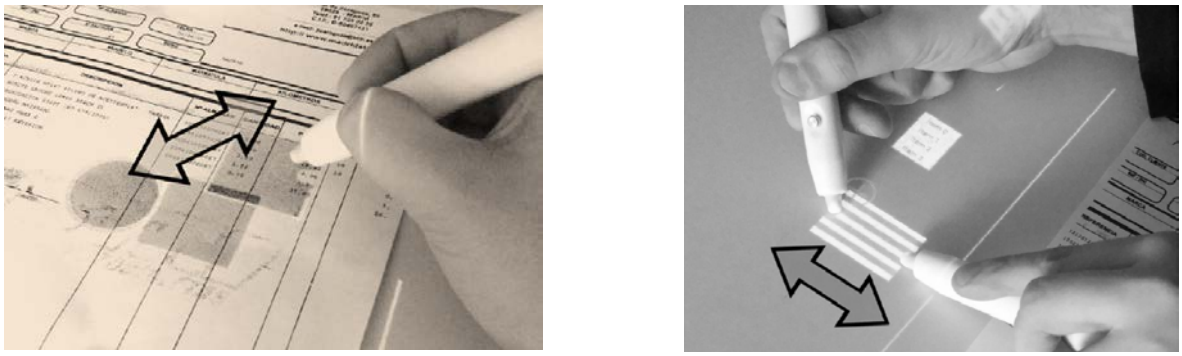

Figure 3. Interaction with the IR-pen system with one and two pens respectively.

For many of the gestures, especially when they last long, the visualization system will provide some visual cue to help the user understand that the gesture has been determined. The different visual cues may go from projecting the input point as a circle, to more elaborated effects such as marking a certain button or menu entry as selected. Since in our case we have implemented a set of widgets that cover the main elements of a virtual desktop, many of these visual cues are implemented as different states of the widgets (e.g. selected vs non-selected). Other effects are simply provided with the interaction of the widgets. For example, when performing a drag-and-drop operation, the element is moved as the user drags its virtual position. This is shown for instance in Figure 3-left. In the first case, a drag-and-drop operation (indicated by arrows on the left) is being carried out by the user. The visual cue that communicates the behavior is the actual translation of the rectangle in purple. We can also perform other two-hands operations such as scaling, as shown in Figure 3-right. The displacement of the pens is also indicated here with the blue arrow, and the user will see an effective incremental resizing of the object while the gesture is not finished.

The gesture management module has been implemented agnostic of the interaction element. We have designed a set of one-hand or two-hand gestures that include simple taps, swipes, and so on, that can be implemented both by hand or IR-pen. In both cases, the user can work with one hand/pen or with two, and the

detected gestures are equivalent for the hand and the pen. The different gestures that are detected when operating with a single hand (tap, scroll, flick, and drag-and-drop) are shown in Figure 4 (4 leftmost gestures). Zoom and resize are easier using two input points. We detect them with the use of two hands (Figure 4 – rightmost images). Input position is determined by detecting a contact between the index and the thumb.

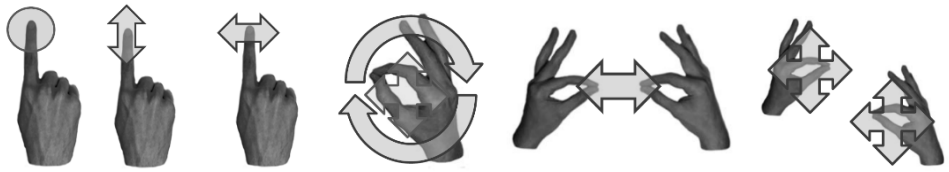

Figure 4. The one-hand gestures that can be detected by our system: tap, slide, flick, and rotate.

4. EXAMPLE SETUPS

We have developed two different setups, **a nomadic** and **a fixed system**. With the aid of the previously described modules, few extra packages were required.

The **nomadic system**, was intended to show the capabilities of a projection-based augmented reality system paired with a camera and with hand or IR-pen interaction. The main features of the system were the portability and the flexibility in mount therefore, several aspects had to be implemented in an adaptive way, which greatly increases the difficulty of the software development. These, were the main decisions we had to take:

- **Hardware:** The hardware, if the system has to be moved, must be light and easy to mount. We chose a small Logitech C615 and a pico-projector from MicroVision, the MicroVision Showwx (an always-in-focus projector), together with an articulated arm that had a heavy basis to be stable on the table.
- **Calibration:** The calibration system may not rely on fixed environment, so several elements such as illumination, background, and so on, must be taken into account in case the system moves.

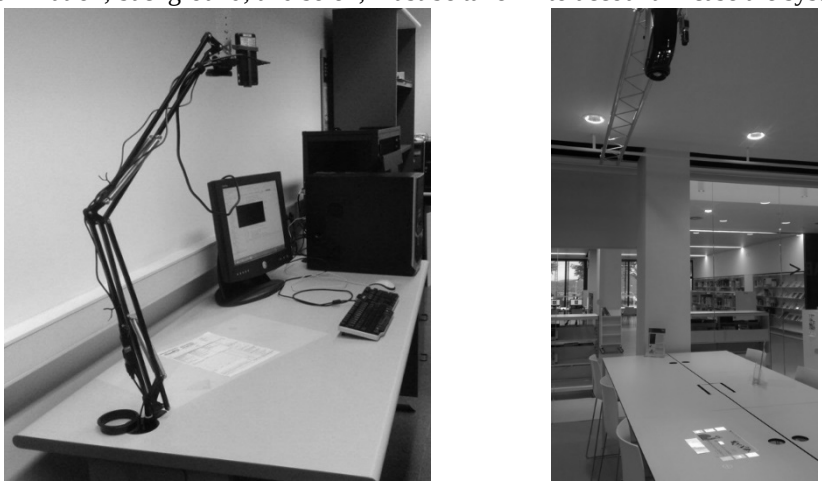

Figure 5. The initial setup of the nomadic system and fixed setup (right) built in a public library. The fixed setup has the projector/camera pair fixed in a structure that is attached to the ceiling. This way, the users have free space around them to experiment and freely perform gestures to interact with the application.

The selected **hardware** with the initial nomadic setup is shown in Figure 6-left. A second version, with a more professional look, was also created with the use of a shell printed using a 3D printer. After a thorough analysis of capabilities of the different devices in the market, in terms of quality, image resolution, distance of projection/capture, and so on. The selected set is affordable and easy to transport. Its total cost was less than $500. For the PC, we used a commodity portable device (less than $1000), with no special features, since the computational requirements of whole system are low. The body of the system consisted on a modified lamp arm and a couple of plastic pieces printed on a 3D printer to fit the camera and projector.

The *calibration* was a second, important issue. Since the nomadic system can be built in different places, we need a calibration process that is able to adapt to different lighting conditions. There are two different aspects (that involve many variables) that may be taken into account when calibrating such a system: a) **Illumination:** Conditions may change between different places, so the calibration system must be as robust as possible to lighting changes, and b) **working area conditions:** The size, color, and orientation of the working area may change due to physical limitations. Although no large room is necessary to fit all the elements, the available space may change from place to place. The calibration can be started as required, since it also uses the same messaging system to communicate the different found matrices, and it uses a background subtraction to increase robustness. The general process follows these steps:

1. Detect the homography between projector and camera
2. Estimate camera parameters.
3. Detect the orientation and size of the valid working area.
4. Calculate the remaining homographies with the working area.
5. (Optional) Calibrate other external gesturing elements, e.g. IR-pen.
6. (Optional) Set-up other input devices, e.g. tablet, that requires some communication set-up.
7. Communicate the homographies.

The calibration stage involves several steps: The first step, where the projector-camera homography is calculated, is achieved by projecting three rectangles with the projector. These three rectangles have different RGB colors and are read by the camera. Then, from these rectangles, the camera parameters are estimated by detecting the rectangles' corners. Next, we use an auxiliary document inside the working area to determine its orientation and size. We determine the maximum valid working space by fitting the largest rectangle in the limits of the camera viewing space and projector space. Once we have the homographies, they are broadcast to the whole system, so that all modules, can read them. From now on, all the communication referring to projected virtual elements, is be carried out in working space. The visualization system is in charge of positioning the elements properly, and automatically repositioning them to avoid occlusions when necessary.

The **fixed system** is composed by a Basler ac2500-14gc camera and a projector InFocus IN 3138HDA, which provides HD projection with 4000 ANSI lumen. Moreover, this system also uses an IR camera, which is basically a very similar camera, a Basler ac2500-14gm with an IR longpass filter (850nm, M27×0,5mm) for the IR Pens. The devices here, in contrast with the nomadic system, can be of higher quality, and the distance of the projector to the surface is of 2.2m, and the area of projection is about 1.1m wide (16:9). The main difference of the fixed system with the nomadic one is the intended use. The objective in this one is to have a living lab in a public library where the users may experiment with projected augmented reality technologies. More specifically, the users will be, mainly, children, and therefore, we have developed a set of small toy applications to be used by the children. In Figure 6-right we can see how the fixed system looks. The fixed system uses the same, previously enumerated software packages to perform all the tasks. The only difference with the previous system is that the projectors and cameras have a larger resolution and can be placed at a larger distance, so that we can build a fixed system that is less prone to accidents.

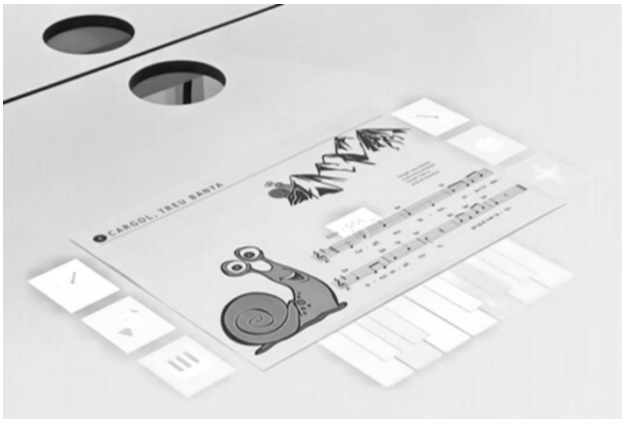

Figure 6. The music toy application where children can play with different instruments.

One of the toy applications we have developed is a music player. In this application, the user shows a music score that the system is able to detect and interpret. The user only has to select the note, and the system plays it. The system can also change the instrument that plays the music by letting the user choosing among a set of predefined instruments. Everything happens in a very user-friendly way, by providing most of the options as icons the users may select. We can see an example of this application in Figure 6, where the projected elements such as the piano tiles or the instrument icons are all interaction widgets. The system tracks the document position, so if it changes, the widgets are automatically rearranged accordingly. The user can choose the instrument, play a note, or play the whole song.

CONCLUSIONS AND FUTURE WORK

In this paper we have presented a software framework tailored for the rapid development of augmented reality setups that are based on the projector-camera pair. The system is highly distributed and all components execute individually and communicate through a communication system based on Google Protocol buffers over a ZeroMQ transport layer. All the modules communicate using a protocol defined by the Communications Module that is the center of all the system. We can even attach external devices (e.g. a tablet) to the communications system. The development of a simple application with our new framework can take as few as a week if no other hardware elements have to be added. It consists basically on reprogramming the *Application Logic* module to fulfill the users' needs. Besides the general modules, we have also implemented other modules for document tracking, widget rendering, and so on, that are easily integrated and can be shared by other modules. In future we want to continue developing the system, but concentrating on new features that may be driven by new example applications, or new input devices.

The authors acknowledge the project support for the project by TIN2014-52211-C2-1-R by the Spanish Ministerio de Economía y Competitividad with EU FEDER funds.

REFERENCES

Bernier, E., et al. 2012. The MobilAR Robot, Ubiquitous, Unobtrusive, Augmented Reality Device. *Biennial Conference on Engineering Systems Design and Analysis. American Society of Mechanical Engineers*, 375–381.

iMatix Corporation. 2007-2012. ZeroMQ Message Transport Protocol. *http://rfc.zeromq.org/spec:23*, checked June 2015.

Google Developers. 2015. Protocol Buffers, *https://developers.google.com/protocol- buffers/*, checked June 2015.

Gomila, L.. 2015. Simple and Fast Multimedia Library.. *http://www.sfml-dev.org/*, checked June 2015.

Hewlett-Packard. 2015. Sprout Official Page. *http://sprout.hp.com/us/en/*, last checked June 2015.

Huber, J., 2014. A Research Overview of Mobile Projected User Interfaces. *Informatik-Spektrum*, 37, 5, pp. 464–473.

Huber, J. et al., 2012. LightBeam: Nomadic Pico Projector Interaction with Real World Objects. *CHI '12 Extended Abstracts on Human Factors in Computing Systems*. ACM, New York, NY, USA, pp. 2513–2518.

Kim, D. 2012. Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. *ACM symposium on User interface software and technology*. ACM, pp. 167–176.

Kolomenski, A., 2013. Realization of a spatial augmented reality system– A digital whiteboard using a Kinect sensor and a PC projector. *Ph.D. Dissertation*. Texas A&M University.

Linder, N. and Maes, P., 2010. LuminAR: portable robotic augmented reality interface design and prototype. In *Adjunct proceedings of the 23nd annual ACM symposium on User interface software and technology*. ACM, pp. 395–396.

Mistry, P. and Maes, P., 2009. SixthSense: a wearable gestural interface. *ACM SIGGRAPH ASIA 2009 Sketches*. ACM, 11.

Sielhorst, T. et al., 2006. Campar: A software framework guaranteeing quality for medical augmented reality. *International Journal of Computer Assisted Radiology and Surgery*, 1 (2006), 29.

Spindler, M., et al., 2012. Towards spatially aware tangible displays for the masses. *Workshop on Designing Collaborative Interactive Spaces for e-Creativity, e-Science and e-Learning at AVI*, Vol. 12.

Tsuji, K. et al., 2013. Robust projection method for a mobile robot with a camera and a projector based on a structured-environment approach-Experiments of the modified method considering a distance between a marker and a robot. *Robot Motion and Control (RoMoCo), 9th Workshop on IEEE*, pp. 36–41.

Weiley, V. and Adcock, M., 2013. Drawing in the Lamposcope. *ACM Conference on Creativity & Cognition*. ACM, pp. 382–383.