A Programming Library for Creating Tangible User Interfaces

Christian C. Ventes, Andrés A. Navarro-Newball and Deivy A. Velasco, Pontificia Universidad Javeriana – Cali - Colombia, Edmond C. Prakash, University of Bournemouth - UK

Abstract— **Tangible User Interfaces (TUI) bring digital interfaces to the real world by using specific devices to achieve a task. They can be more intuitive, allowing the user to take advantage of a computer tool which is associated to the real world. One problem is that creating a TUI for each piece of software is expensive. For instance, devices such as the mouse, keyboard or touchscreen have become more popular. Indeed, it is cheaper to adapt the users to the interface than creating an adequate interface for each program. We present VirtuaOM, a library which allows creating low cost interfaces where the users can communicate with an application in a tangible manner. Additionally, an application using this library can allow several users to communicate collaboratively among them and with the system within the interaction space. In order to build our library, we combined the Design Thinking and Software Engineering methodologies. We tested VirtuaOM creating an interaction space inspired in the Sensetable device developed by Patten [9] that permits programmers to create applications where the system can track users and tangible wireless objects in a tabletop surface, but we moved the interaction area from a table to the floor to increase it and to give users the freedom to move through it. This made it easier for multiple users to interact with each other and with the system collaboratively.**

***Keywords*— Application programming interfaces, Augmented Reality, Tangible User Interfaces, Tabletop**

I. Introduction

Interfaces should adapt to the users' multimodal nature in order to create different interactions to perform diverse tasks. Also, they should be capable to work with multiple users collaboratively. Graphical User interfaces (GUIs) only display data on a screen based on pixel information [2, 11]. However, interaction through screen pixels may be inconsistent with the interaction of the user with the physical environment. Actually, GUIs do not take advantage of the human skills to manipulate different objects and data and are limited to the screen. In contrast, Tangible User Interfaces (TUI) take advantage of human's ability to have haptic interactions providing physical representations of the digital information. These representations become a model and controller for the information and allow direct manipulation and multi sensorial interaction. TUIs offer the capability to work in the real world and to model gestures and actions based on the application's context. Also, they facilitate cooperative interaction and profit from the human ability of making better associations with tangible elements [2, 6, 7, 8].

However, a risk found in the development of tangible interfaces is the need of building a specific tangible device for every software made. In some cases, it is not even possible to represent physical and structural changes in those devices [11], as a consequence, the product cost increases. Even if mouse and touchscreen interfaces cannot bring digital information to the real world, they are very popular because they provide a generic way to interact with software without performing changes in hardware but creating new gestures based in a small set of actions provided by those interfaces (e.g. point and click using mouse or multiple contact points in touchscreens). Given this fact, a good idea may be creating a tangible interface with a set of actions that users can perform and the possibility of creating new ways to interact based on that set. Nevertheless, the amount and the difficulty of actions performed is something important to have in mind at the time of designing the interface. We believe that if a user recognizes actions in a device it could be simpler for him or her to infer new actions [10], so our task was to find or design a tangible interface where users could perform a small set of actions; these actions must be intuitive and make sense to the user; the less he or she has to remember the better. Also, in case that it is needed to create new gestures, those gestures should be based on that small set of actions.

II. Related Work

Our work is based in the TUI literature, we explore how to achieve multimodal and collaborative interaction between users and with the system, so we explore several examples of TUI to verify if there are any systems capable to fulfil those requirements. First, we explore the gestural interface approach as a way of natural and multimodal interaction.

A. Gestural interface

G-stalt allows interaction using special gloves to detect gestures. Those gestures let the user organize and view different videos in a 3D space, which implies that the system must recognize different gestures to do different actions [8,

C. C. Ventes is with the DESTINO research group at the Pontificia Universidad Javeriana, Cali - Colombia (e-mail: ccventes@javerianacali.edu.co).

A. A. Navarro-Newball is with the DESTINO research group at the Pontificia Universidad Javeriana, Cali - Colombia (corresponding author; phone: 572-3218200 - 8914 ; e-mail: anavarro@javerianacali.edu.co).

D. A. Velasco is with the DESTINO research group at the Pontificia Universidad Javeriana, Cali - Colombia (e-mail: dandresv@gmail.com).

E. C. Prakash is with the IRAC research institute at the University of Bournemouth, UK (e-mail: eprakash@bournemouth.ac.uk).

DOI: 10.5176/2251-3043_4.1.304

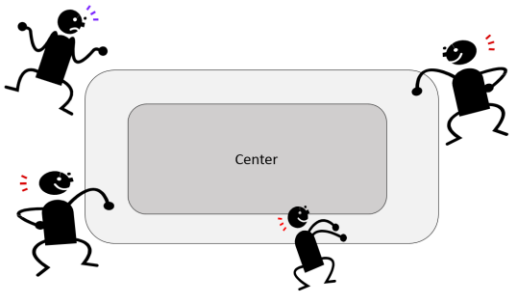

Fig. 1. It is difficult to reach the center in a large tabletop surface.

12]. However, several challenges come when we try to create a gesture grammar capable of recognizing actions made by a user to interact with a system.

B. TUIs

TUIs have been long used [1] and their advantages over GUIs have been evidenced [20]. Lapides, Sharlin and Sousa [21] describe a TUI which follows an intuitive drawing board metaphor to map the 3D task to a physical interface. However, this method is aimed for controlling a group of robots in three dimensional (3D) space only. Costanza et.al. [1] describe a low cost TUI, downloadable from the Internet, where the physical interactive objects are made by gluing printed labels onto existing objects or, are folded out of ordinary paper, with visual markers printed on it. The system only requires a regular computer, a webcam and a printer. However, it is aimed only for musical applications.

C. Multipurpose TUI toolkits

Several toolkits support the implementation of functional TUI prototypes [22]. These toolkits lower the threshold for implementing functional TUI prototypes by handling low-level events. However, most of them provide support only for specific technology. For instance, if different technology is used, developers are required to learn and rewrite new code. Additionally, toolkits do not provide a comprehensive set of abstractions for specifying and discussing tangible interaction. Indeed, TUI design and development has conceptual, methodological, and technical difficulties [22] due to the lack of appropriate interaction abstractions, the shortcomings of current software tools and the excessive effort required to integrate novel input and output technologies. Shaer et.al. [23] introduce the Token and Constraints (TAC) paradigm for describing and specifying tangible user interfaces (TUIs). The paradigm enables the description of a broad range of TUIs by providing a common set of constructs. However, the paradigm neither considers TUIs' interoperability nor TUIs' implementation. Shaer and Jacob [22] extend the previous idea proposing a specification paradigm for designing and implementing TUIs that uses high-level constructs which abstract away implementation details and can be used by TUI developers from different disciplines. The system is actually being extended to support whole body interaction by specifying expressive movement. Among the main challenges observed in TUI design and implementation [22] are:

1) Deciding which information is best represented digitally and which is best represented physically.
2) Inventing metaphors that give physical form to digital information.
3) Defining a behavior for each possible context of use for each interaction object.
4) Standardizing I/O devices for TUIs.
5) Using a low-level programming TUI to deal with continuous interaction.
6) Considering issues such as access points, as well as spatial and temporal coordination for parallel interaction.
7) Providing languages and tools to TUI developers from different disciplines.

D. Tabletop surfaces

Among TUIs, there is a set of interfaces called the tabletop surfaces. There is some evidence that tabletop surfaces can enhance collaboration, exploration of experimental data and learning [24, 25, 26]. For example, Of BATS and APES [27], an interactive tabletop system for natural sciences museums, favors collaboration and discussion among visitors and allow them to achieve an active and prolonged engagement which incentivizes research, observation and construction of knowledge. Of BATS and APES also takes advantage of videogame concepts and design to enhance interaction. However, these tabletop surfaces are limited to actions performed in the 2D space which comprises the interface. Other tabletop systems extend the interaction to the 3D space. For example, eLabBench [28] is a tabletop system which allows biologists to organize their experiments. Here, biologists can pull digital resources, annotate them, and interact with hybrid (tangible and digital) objects such as racks of test tubes. Nevertheless, when the interaction is extended to the 3D space challenges with occlusion arise. In ObjectTop [29], a system to support tabletop display applications involving both physical and virtual objects, the challenges of occlusion by physical objects are addressed and the advantages of occlusion-aware techniques are shown.

E. Multiple purpose tabletop interfaces

We found some examples of tabletop surfaces created for multiple purposes. Those interfaces had their own API to allow users to create new different types of software based on the limited set of actions that those interfaces provide. The popularity behind the tabletop as multi – purpose tools could be based in the capability of having a large area to organize objects spatially. Spatial arrangements are integral part of humans' behavior; they simplify the way we perceive our world, our choices and they help us through our internal computation [13]. Examples of tabletop surfaces that allow user creating their own application or interactions include Pingpong++ [15] which is an augmented ping pong table capable of detecting when users touch it and show a visual response projected on its tabletop surface with its own *API* to create visualizations. Additionally, tabletop devices such as trackmate [14], Sensetable [3] and reacTIvision [16] interact with the system through tangible devices above the surface instead of touching it. This gives more possibilities of interaction other than only contact; with this, the system now

can track object position and orientation of multiple objects which can be organized in a logical way to solve a problem or become physical embodiments of digital information. Improving even more, ZeroN is a tabletop surface capable to re arrange tangible elements in a 3D space through magnetic levitation [17].

F. Enlarged interaction area

An increase of the interaction area is achieved with systems such as The Magic Carpet [18] where the sensors used detects people kinematics to track dance movements. This system could keep away most part of the sensors used, making it a promising tool, but we have to keep in mind that this system was not made to detect and differentiate people and tangible objects. Also, the magic pad; a spatial Augmented Reality based interface can track the position and orientation on a pad inside a CAVE type interface [19]. The system detects the pad position and projects visual information on it. The problem with the tool is that it separates the user from everything outside the CAVE. We are looking for something easier to install and that "blends" with its environment.

III. An affordable tangible tabletop surface for the floor

After researching different examples of tangible interfaces we paid special attention to the Sensetable device developed by Patten [3, 9], due to its device interfaces which consist in small tangible devices similar to Hockey pucks working above a tabletop surface [3, 9]. Those devices only had two possible movements: rotation and translation which are easy to perform and remember. Also, in case of needing more functionality it is possible to create more different types of interaction based on those actions such as semantic zooming dragging two pucks closer or farther, or attach/release projected visual information in the surface to a puck after performing some action with it. The Sensetable has its own API that allows programmers to access to every puck data; this can be used to extend the puck functionalities.

If we think in cases such as museum exhibitions where technology should be a reinforcement of the current material, not a replacement; we want to increase the interaction area while not allowing the tangible interface to become the center of attention. On the contrary, we expect that the interface "blends" with the exhibition, allowing users to interact with the museum material in new ways through the technology. At the same time, we expect that our interface can be used by more than one user, allowing collaboratively interaction between users and the system. After thinking in several possibilities for increasing the interaction area of the Sensetable or any other tabletop surface, we can observe that if we translate that area to the floor this will not disturb the movement of people through the hall where the interface is located. Also, if a larger Tabletop surface is used, the larger its interaction area is, the harder will be for users to move any tangible device to the center of that area (Figure 1). Although translating the interaction area to the floor seems promising, it has a very important drawback. If the interaction area is increased, the tangible elements (pucks) should be bigger. Thus, we need to find new ways to detect and track those elements. Also, if users are able to enter the interaction area, the possibility of damage or loss of the elements of the interface (detection sensor and tangible elements in case of the Sensetable) will increase, increasing with this, its maintenance cost.

In our approach, we seek to achieve the same detection of tangible objects in a larger tabletop area located in the floor but keeping away any electronic device from the detection area or the tangible objects. With this we prevent multiple users from damaging the interaction area. Also, tangible objects are easier to replace in case of damage or loss. After researching several methods for object tracking we decided to use the Kinect sensor due to its capability to use several sensors of different type supporting each other for a better detection.

With the location of the detection area in the floor a user now can enter the interaction area. Our system contributes with new interactions between a group of users and tangible objects. The result is VirtuaOM a library that permits creating applications where multiple users can interact with tangible objects in a $4m^2$ area. This Library has a set of instructions that allows detecting position and orientation of those tangible objects, detecting users who interact with those objects and giving the possibility of improvising new type of interactions. The use of a Kinect sensor and having no presence of electronic devices on the tangible objects allows that those new interactions can be developed in an easy and economic way. We describe the methodology used to design and implement this system, also the results obtained after its implementation and new opportunities for further work.

VirtualOM proposes a solution for challenges (2), (3), (4), (5) and (6) described in Section II, C . The implementation of VirtuaOM using a game library allows the developer to invent elaborated metaphors and narratives (2) to provide information and to define different behaviors and continuous interaction (5) taking advantage of the update loop from the game and the object oriented paradigm (3). The use of the Kinect sensor minimizes the use of electronic I/O devices, "standardizing" interaction with real objects and; aids parallel interaction in conjunction with an augmented reality toolkit (6). Additionally, we present a solution to cope with occlusion problems and provide a mechanism to compose interaction gestures from basic ones.

VirtualOM is not a specific application [1, 21], but rather a multipurpose interface creation tool [23], but with no specification language associated. Its technology can allow the easy implementation of video game narratives [27]. Additionally, it allows the developer to extend to the 3D space [28] but in a larger area, taking advantage of TUI and tabletop features [18, 19], but including any real object.

IV. Methodology

We combined two approaches for problem solution. Design Thinking is typically far from Software Engineering methods and it is mainly used in Human Computer Interaction for interface design [30], which typically runs separately from Software Engineering processes. However,

there is evidence of successful use of Design Thinking aligned to Software Engineering [31].

Design Thinking is a method for designing user centered solutions [32]. In Design Thinking there are three design spaces: Inspiration, which describes the problem or opportunity motivating the search for a solution; Ideation, which is the process of generating and testing ideas and; Implementation, which is the execution of the ideas. These spaces are iterated several times until the wanted solution is found. Generated ideas are validated within the workspaces according to project restrictions which consider criteria such as feasibility (is it functional?), viability (is it sustainable?) and user desirability (does it make sense for the people?)

Similarly, in Software Engineering there are stages which are iterated to transform a set of requirements into a solution. Particularly, in Agile Development, software evolution is constantly presented and discussed among users, iterating an incomplete prototype until the user is satisfied [33]. It has similarities with Design Thinking. However, there is no way to communicate conceptual decisions to final users [31]. Thus, we used diagrams or analogies to present ideas (e.g. Figures 1 and 5A) and used rapid prototyping and participative design from Design Thinking embedded in Agile Development. Additionally, we performed the activities in each design space to find a solution, but as new options were explored, we used mathematics, semantics and modeling tools to describe a solution. In order to avoid excessive repetition for each solution option we followed a waterfall approach derived from software engineering, where each step involved Design Thinking Activities. We ended up with the following stages and tasks for the whole design process:

Analysis. *We included all tasks related to the Inspiration space. Here, we defined the technical details and some non technical ones, but no design details were given. It included the following tasks: audience definition, prioritization, success criteria, glossary, problem history, state of the art, expert opinion, features selection, available technical options.*

Design. *We included all tasks related to the Ideation space. We explored different alternative solutions, specifying imitations and advantages and, choosing one at the end. It included the following tasks: establishing systems´ operating conditions, general modeling, modularization, input device design, defining input and output operations, acceptability criteria definition, algorithm design, output GUI design, test experiment design.*

Implementation. *We included all activities in the traditional test and implementation method once a solution is found. Here, it is necessary to iterate to find enhancements or to solve problems. Once finished, the solution is executed.*

Sections V, VI and VII mainly explain the solution alternative chosen following this process. However, when required, we explain other options. We applied this method to different parts of VirtuaOM at different development stages. An earlier version was presented at the CGAT 2014 [4].

V. ANALYSIS

VirtualOM was conceived given the necessity of creating "transparent" or "invisible" (non-intrusive) interfaces that can guide multiple users' interactions within a specific environment (e.g. museum exhibitions). Devices like the Sensetable may be incorrect when you want to use the technology in a room with a considerable number of users. A naive solution could be placing several devices in different locations and relating them to the actual context. However, this requires to modify infrastructure, which can be invasive to the environment where the tabletop surfaces are located. We may enlarge the device to obtain a bigger workspace; however, this option raises maintenance costs and requires the use of a big table or surface where the device would be placed (occupying much of the real available space) breaking with the idea of a transparent (invisible) interface.

VirtuaOM proposes a floor based detection workspace with the following features:

1) Large enough to allow collaborative interaction of many users (at least four square meters).
2) Avoids the need of modifying the existing floor.
3) Physical interaction elements (pucks) that do not rely on electronics, thus, lowering maintenance costs.
4) Allow displays of virtual information through any projector or screen, thus, floors which are not suitable for projection are not a limitation.

B. New Tangible Objects

For an initial prototype we decided to establish the interaction area to $4m^2$. If we decide to keep tangible objects with the same size of the tangible objects of devices like Trackamate or Sensetable, it will be really hard to track and locate them. Also, real interaction will be done by the user's body. This is the scenario where user has to remember several instructions and there is a chance of some of the actions being misinterpreted by the system giving us incorrect results. Some devices are able to detect translation, pitch, and roll and could achieve an easier detection but are be dependable of electronics, which increases their maintenance cost.

The option we decided to implement was increasing the size of the pucks. Large enough to allow a user to manipulate them with both hands and so that the top of them could be located at the height of an average user waist. With tangible object of that size we could add two new type of interactions: multiple users grouped in a single puck and users using a puck as a reference (Figure 2).

C. Considerations About Detection Latency

Patten [3, 9] establishes that one of the greatest strengths of the Sensetable is the use of the electromagnetic approach for object tracking. He says that the method used in the three prototypes of the Sensetable is particularly faster and less susceptible to failure than computer vision approaches [3].

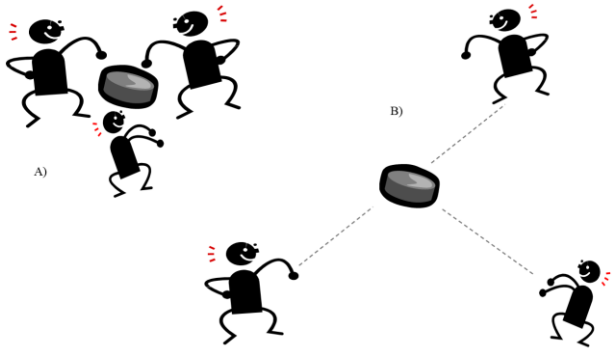

Fig. 2. Adding two new interactions with larger pucks. A) Multiple users in a single puck. B) Using a puck as a reference.

Independent of the method used to track the tangible objects it is important that our interface is capable to detect pucks in real time. But how "fast" the sensor used needs to be?

The first prototype of Sensetable uses Wacom tablets as detection sensor [20]. In order to compare, we searched for a Wacom tablet's estimated latency; we found that WACOM STU-300 Stylus can achieve a latency of 200 detections per second [20] which is hard to surpass. Nevertheless we must remember that to give a human the perception of animation, it is only needed to show 24 images per second approximately and computer monitors normally refresh at latency of 60Hz. It is true that the lower the latency the better, but if the selected approach can reach between 25 – 60 frames per second, we believe that is enough for making detections in real time. Using a sensor capable to give 200 detections per second could be more than is really needed.

D. Kinect as Detection Sensor

The Magic Carpet uses more than one sensor in order to improve its detection. Specifically, it uses piezoelectric sensors to detect the user position and Doppler Effect sensors to determine user movement [18]. We selected the Kinect device as a sensor because it is composed of different Sensors that work using their own SDK in order to make easier the detection of users. Additionally, it can be used in a distance where users cannot reach it when they are being detected by the system [21]. The Kinect can provide information through 3 different type of sensors:

1) Images without processing or compression through a CMOS RGB camera.
2) Object depth information through a CMOS monochromatic infrared camera.
3) Audio Data through 4 microphones that organize the sound in frequency arrays.

Kinect cameras possess an optimal area (sweet spots) where it is assured that the system will work with less failures inside their physical limits which is the maximum area where the Kinect can operate. Also Kinect operates in two modes. Near mode to position Kinect at closer distances and Default mode which permits working at larger distances but is more susceptible to noise. Finally, the Kinect's camera has a horizontal angle of vision of 57.5° and a vertical angle of 43.5°; also, it is possible to tilt the Kinect 27 degrees upwards or downwards[21].

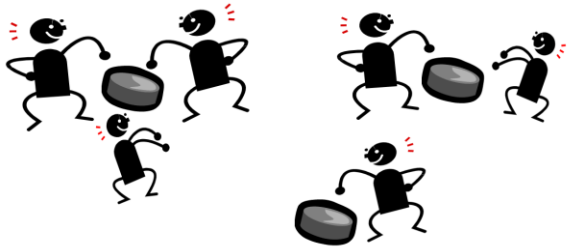

Fig. 3. Recommended user interaction.

The Kinect microphone is programmed to recognize human voice sounds and small sounds produced by human body movements. It can detect audio signals from 50° to the left and right of the sensor. Virtually, the microphone could be divided in 10° arrays that can be used to detect the source of the sound.

Kinect's microphone is specially developed to detect noises from the voice or human body movements, but it is hard to know if it can be useful to detect tangible objects. On the other hand, it is easy obtain data from the RGB and depth cameras. That information could be used to differentiate or track users and pucks using computer vision methods. An advantage of using the depth camera is to be able to work under low light conditions. Given the fact that depth and stream data are relatively similar, both can be used together to create our tracking approach.

VI. Design

Our task is to build a system where users interact with tangible elements (pucks) located in a floor area. Some considerations we had in mind when we designed our interaction were:

1) The system must be capable to distinguish pucks from users. Also it has to be capable to detect when a group of users are near a puck.
2) Users can apply two valid movements to the puck. Translation over a detection area and rotation of a puck in its own axis.
3) Puck height should not surpass the height of the user waist and should be big enough to be manipulated by a user with both hands. Also, they cannot be too heavy for being picked up and moved through the area.
4) Sensors must not become an obstacle for the elements inside the interaction area.

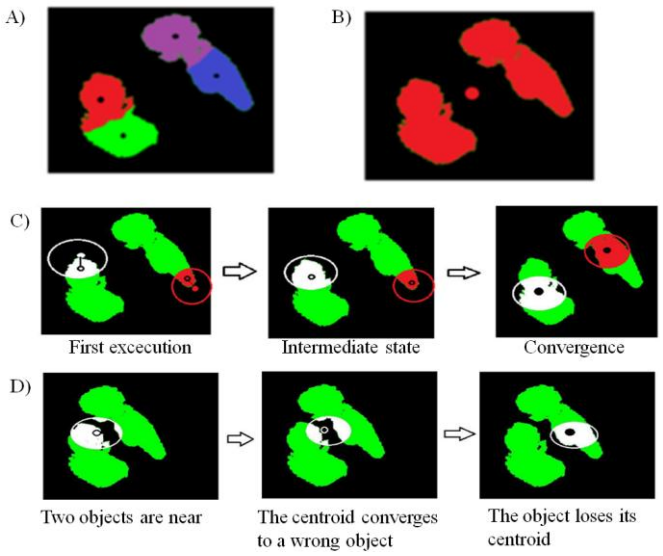

Fig. 4. Image segmentation with Kinect's depth sensor. A) Centroids. B) Incorrect centroid selection. C) Circular windows to limit centroid selection. D) Wrongly assigned pixels.

According to those consideration, the system can recognize two types of actors interacting on the system. Those are users and pucks. We had to keep in mind that if there are too many of those actors inside the area, the space to operate will be reduced. Also, it doesn't make too much sense if all users are grouped in a single puck. For that reason we recommend limiting the number of pucks to 3 per area, also, the system only should detect when 3 users are near a puck. Others should be ignored (Figure 3).

B. Design of the Detection Method

The depth stream stores every pixel of the Kinect's infrared camera. Different to the RGB camera, instead of storing color information, these pixels store distance information that can be obtained using the Kinect SDK. If we ignore every pixel not located at a defined distance, we can make image segmentation to detect objects that enter into the area (Figure 4). However, image segmentation helps detecting elements in a specific range but it cannot identify all the actors that enter the area. To solve this we use the mean – shift algorithm to cluster the pixel data [5, 22], this allow us to create centroids to identify every element which enters in the detection area (Figure 4A).

Nevertheless, an incorrect selection of centroid could show incorrect clustering. Too much centroids could separate a single actors in several objects, a very few will take several actors as if they were only one (Figure 4B). Another problem found with the mean shift algorithm is the incorrect assignment of pixels to a centroid. In this algorithm every pixel is assigned to the nearest centroid but depending of the actor shape there are cases where a pixel of some actor is closer to another actor. To solve this, we implemented circular

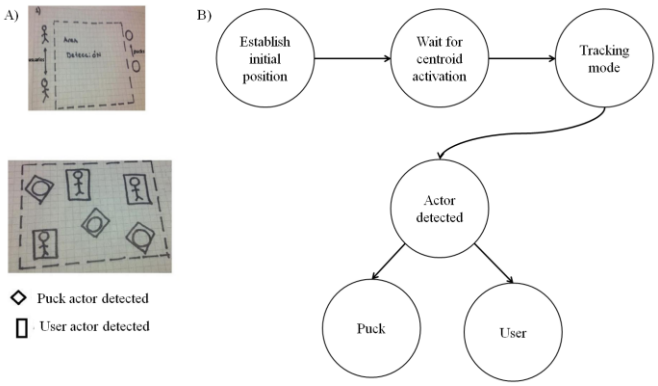

Fig. 5. Actor identification scheme. A) Sketches to communicate with designers. B) A portion of the finite state machine used to communicate with software developers.

windows to limit the number of pixels that are assigned to a centroid. Now only those pixels inside a window are the only ones assigned to a centroid (Figure 4C). Once every possible pixel is assigned, a new centroid is calculated using the pixels inside every window. The algorithm is repeated for every depth camera refresh and at some point the centroid converges and the objects are identified. The last problem found is when two objects are too close. When that happen pixels belonging to an actor could be inside the window of a second actor. After several iterations those wrong assigned pixels can make the windows located in the first element converge in direction of the second element with an incorrect tracking as a consequence (Figure 4D). To solve this we implemented a collision system where a collision occurs when two virtual windows intersect each other. In our method, when two centroid windows intersect, we detect a collision. In that moment instead of recomputing the centroid, it remains the same until there isn't a collision anymore (the actor moves).

C. Detecting Puck Orientation

We use the Kinect RGB camera to detect puck orientation. Specifically we took advantage of a concept from augmented reality to achieve this. Augmented reality software can use special images called glyphs or markers. Those images become a reference to a camera where the 3D virtual images should be displayed in order to combine both real and virtual worlds in a single presentation. The matrix (1) performs the conversion of the coordinates of the glyph according to the world to camera coordinates (X_C, Y_C, Z_C). The Matrixes V and W contain the rotation and translation performed in the glyph in the real world in computer coordinates. Specifically, if we attach a glyph to the puck, we can use the matrix V to obtain object orientation.

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = \begin{bmatrix} V_{11} & V_{12} & V_{13} & W_X \\ V_{21} & V_{22} & V_{23} & W_Y \\ V_{31} & V_{32} & V_{33} & W_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_M \\ Y_M \\ Z_M \\ 1 \end{bmatrix} = \begin{bmatrix} & V_{3x3} & & W_{3x1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_M \\ Y_M \\ Z_M \\ 1 \end{bmatrix} \quad (1)$$

D. Actor Identification Scheme

The mean-shift algorithm is useful to detect and track multiple actors, but it is not possible to identify which actor is a user and which actor is a puck. We designed a scheme where the programmer decides how many users and pucks the application will have, the order of how they will enter to the area and their respective initial positions. Initially some initial positions have to be established. When an application is running it has 2 modes (Figure 5). The detection mode when the system waits until all centroids activate. An activation occurs when we place an actor in the initial position. Every actor is activated one by one and there is no chance that both actors are activated at the same time. There is no problem with how users should enter, but every puck has to be activated in its initial position. A puck in a wrong initial position cannot be allowed. To help the system identify pucks in the system we use the same Glyphs used to track their orientation.

After every puck is activated, the system enters to tracking mode, where the system starts tracking all the actors' position. It is required that at least a user stays near a puck in order to move. This is because it has no sense that a puck moves by itself. Also, in this stage the system has to know when a group of users is near a puck.

VII. Implementation

A. Sensor Position

Now that we have an actor identification scheme working with the mean – shift algorithm, the next thing to establish is the sensor position. We tried three possibilities; first, the frontal position which is completely different to the tabletop notion and has problems detecting actors when an actor is behind another. Next, we have the top central position, where the Kinect is located in the top – center of the area probably in the ceiling. The problem with this option is that we discovered we needed to position the Kinect at least 2.76 meters at top to reach a 4 square meters area. Finally, we have the diagonal position. In this configuration the Kinect is at the top too, but not in the center. With this configuration it is possible to cover the 4 square meters required using a height less than the top – center configuration. In Figure 6 we can see the Kinect in a diagonal configuration. The darker part is the detection area, actors will be detected at that area. The lighter area is called the death zone and is beyond the Kinect view angle, this means this is the minimal horizontal distance where actors can be detected. That zone depends of the height where the Kinect is located. If we position the Kinect at 2.72 meters but using diagonal configuration instead of top center. We will have a death zone of 0.78 meters, also we could reach an area of more than 5 square meters. The problem with it is that the more height the less visibility, so we can't position too high. Our test were made at 2.26 meters and we covered the 4 square meters with no problem. Calculations are explained in Equations (2)(Figure 6):

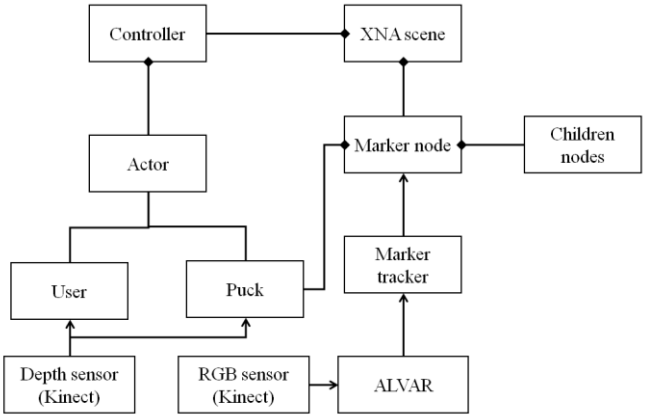

Fig. 7. VirtuaOM Architecture.

$$
\begin{aligned}
h &= \frac{2.72m}{\sin(70.5)} = 2.89m \\
b &= \sqrt{h^2 + 2.72^2} = 0.78m \\
a/\sin(A) &= h/\sin(B) \\
a/\sin(43.5) &= 2.89/\sin(27) \\
a &= 4.38m
\end{aligned}
\qquad (2)
$$

B. System Architecture

We needed a system capable to integrate the Kinect with some augmented reality library in order to use both systems at the same time in an easy way. Goblin XNA is an open platform to create 3D user interfaces with XNA. We decided to use XNA because it uses XNA functions that can be easily integrated with the Kinect SDK developed by Microsoft. Also Goblin XNA simplifies some of the normal functions from XNA by the use of different type of nodes making easier work with 3D graphics and augmented reality and to include videogame narratives taking advantage of the XNA game library.

VirtuaOM is used in conjunction with XNA, so it is mandatory to install Goblin XNA, to create a XNA scene and call both Goblin XNA and VirtuaOM to be used. Our system has a main Controller class where after the programmer creates all the actors he or she wants, they will be stored on that class (Figure 7). Those actors, no matter if they are Users or Pucks depend on the Kinect camera. On the other hand, the Puck class uses the marker node class which is the class that manages the augmented reality Glyphs in Goblin XNA. That class depends of a library named ALVAR which is the class which really is in charge of the augmented reality modules and at the same time the class ALVAR depends on the Kinect. The code required to use the actors includes:

1) Actor and controller definition.

public puck.puck marker1; // puck actor

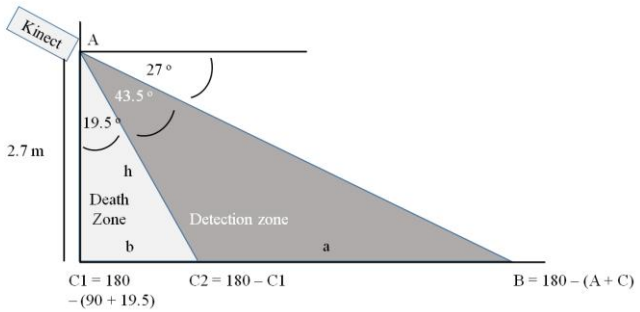

Fig. 6. Kinect position.

public puck.persona persona1; // user actor
public puck.controlador referi ; // controller

2) Actor and controller initialization.

marcador1 = new puck.puck (x,y,z, 1); // puck actor
person1 = new puck.persona (x,y,z, 0); // user actor
referi = new controlador(); // controller

3) Adding actors to the controller.

referi.Add_element(marcador1); // puck actor
referi.Add_element(persona1); // user actor

C. Independent centroid calculation every Frame.

At first, when we executed the mean-shift algorithm in some computers it was impossible to do real – time detection. This happened because the mean shift calculation requires a lot of calculation when there are more than 3 authors or more at the same time. There is no problem when we use the algorithm with a single author. We present a case with a Core I3 computer with no specialized graphics card, where we test the system under 2 conditions. In the first one, we disabled the window collision calculation, in the second we ran the system under normal circumstances. In the first case we obtained a 29FPS latency, with the second we obtained 14FPS latency. We figured out that calculating a single centroid for tracking is so fast that the human eye cannot see it, so we decided to calculate every author separated in a different cycle. The number of calculations made every refresh decreased. With this we achieved a 26FPS latency doing all the calculations.

VIII. Results

We evaluated the response time in 3 different computers with their own characteristics under 2 experiments. The first experiment consisted in 3 users and 3 pucks moving around the detection area. On the second experiment, we asked the users to change pucks constantly and to form groups on every puck. The computers used in the experiment were core i3 processor pc with a low performance graphic card (PC1), a core i5 with an Intel HD 4000 graphic card (PC2) a high performance core i7 with an NVidia GTX460 graphic card (PC3). The results obtained are shown in Table I.

In the experiments, we noticed some fluctuations on the average latency. Big part of the low latency is obtained when 3 or more authors are together. Nevertheless, with the fluctuations, it is still possible to obtain detections on real time even with low performance computer.

Additionally, to show the capability to integrate VirtuaOM with other multimedia libraries and to show the possibilities on multimodal interaction we developed a tangible audio reproducer. On this application we used the library irrKlang to control sounds. The idea is that 3 users control 3 pucks to do the next actions (Figure 8):

1) Position puck 1 up in a play button to play/pause a song.
2) Walk through a specific area with the puck 2 to change and repeat a song position.
3) Rotate the puck 3 to change the volume of the song.

Then, we evaluated the interface with users based on the Nielsen design heuristics [10]. We discovered that the interface was easy to understand by users but it fails with user control and freedom because when a user goes out from the detection area there is no way to show that what he or she is doing is wrong.

Also, we asked five software developers how easy they felt the library was and they all agreed if was very straight forward to program. Finally, we asked five potential users from one museum for potential application of this library and they found it very useful.

IX. Conclusion

A benefit of implementing the library working together with Goblin XNA is that the use of this platform allows to present different type of multimedia content. The children nodes presented in Figure 7 are used by Goblin XNA to associate to main nodes, different types of multimedia files, such as sounds, 3D images, 2D images and videos. We can exploit that feature to improvise new type of interactions with VirtuaOM. Additionally, the augmented reality interface allows VirtuaOM to show information in different ways than a projection. This is important because not all floors can be used for projections.

We recognize the importance of the tangible interfaces in order to allow the user a multimodal interaction with a computer, also the importance of having an interface that allow us interacting not only with the system but with other users. VirtuaOM was built as an alternative to create tangible collaborative interfaces of multiple purpose without being too expensive not only on production but in maintenance cost.

TABLE I
Response Time Evaluation

PC	Experiment 1	Experiment 2
PC1	12 - 27 FPS	17 - 27 FPS
PC2	40 - 50 FPS	27 - 30 FPS
PC3	60 FPS	60 FPS

VirtualOM proposes a solution for challenges (2), (3), (4), (5) and (6) described in Section II, C.

The computer vision approach allowed us keeping the sensors in places where they do not disturb the user when he or she interacts with the system. Even if other approaches are faster to detect and track objects, we showed that is possible use computer vision at 25-60fps latencies, which are enough for real time applications.

The best field of action on VirtuaOM are interfaces when it is necessary create an interface capable to interact not only with users collaboratively but with the environment where is located. In places like museums exhibitions, where the computer system should not make the users ignore the museum content, on the contrary it should become a guide for the users through the installations, interfaces like VirtuaOM could be a great option, especially if we take advantage of the integration with Goblin XNA or the possibility of using other libraries to create Augmented and Mixed reality options.

REFERENCES

[1] E. Costanza, M. Giaccone, O. Kueng, S. Shelley, and J. Huang (20100. Tangible interfaces for download: initial observations from users' everyday environments. In CHI '10 Extended Abstracts on Human Factors in Computing Systems (CHI EA '10). ACM, New York, NY, USA, 2765-2774. DOI=10.1145/1753846.1753862 http://doi.acm.org/10.1145/1753846.1753862

[2] H. Ishii, B. Ullmer (1997). "Tangible bits: towards seamless interfaces between people, bits and atoms". In Proceedings of the ACM SIGCHI Conference on Human factors in computing systems (CHI '97). ACM, New York, NY, USA, 234-241. DOI=10.1145/258549.258715 http://doi.acm.org/10.1145/258549.258715

[3] J. Patten, H. Ishii., J. Hines, G. Pangaro. (2001). "Sensetable: a wireless object tracking platform for tangible user interfaces". Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '01). ACM, New York, NY, USA, 253-260. DOI=10.1145/365024.365112 http://doi.acm.org/10.1145/365024.365112 .

[4] C. C. Ventes, A. A. Navarro-Newball, D. A. Velasco, E. C. Prakash, (2014). " VirtuaOM: Tangible Human-Computer Interface for Collaborative Applications." . En: Singapore Proceedings of the 7th Anual International Conference on Computer Games, Multimedia and Allied Technologies (CGAT 214), 25-33 ISSN: 2251-1679.

[5] P. M. Roth, M. Winter. "Survey of appearance-based methods for object recognition". Published January 15 2008. Inst. For Computer Graphics and Vision Graz University of Technology, Austria 134

[6] B. Ullmer, H. Ishii, R. J. K. Jacob (2005). "Token+constraint systems for tangible interaction with digital information". ACM Trans. Comput.-Hum. Interact. 12, 1 (March 2005), 81-118. DOI=10.1145/1057237.1057242 http://doi.acm.org/10.1145/1057237.1057242

[7] K. Nesbitt, B. Orenstein, R. Gallimore (1997) "The Haptic Workbench applied to Petroleum 3D Seismic Interpretation". The Second PHANToM User's Group Workshop, 1997.

[8] M. Blackshaw, A. DeVincenzi, D. Lakatos, D. Leithinger, H. Ishii (2011). "Recompose: direct and gestural interaction with an actuated surface". In CHI '11 Extended Abstracts on Human Factors in Computing Systems (CHI EA '11). ACM, New York, NY, USA, 1237-1242. DOI=10.1145/1979742.1979754 http://doi.acm.org/10.1145/1979742.1979754

[9] J.M. Patten (1999). "SENSETABLE: A Wireless Object Tracking Platform for Tangible User Interfaces". Master dissertation thesis, University of Virginia.

[10] J. Nielsen (2013). "10 Usability Heuristics for User Interface Design". http://www.nngroup.com/articles/ten-usability-heuristics/

[11] H. Ishii, D. Lakatos, L. Bonanni, JB. Labrune (2012). "Radical atoms: beyond tangible bits, toward transformable materials". interactions 19, 1 (January 2012), 38-51. DOI=10.1145/2065327.2065337 http://doi.acm.org/10.1145/2065327.2065337

[12] J. Zigelbaum, A. Browning, D. Leithinger, O. Bau, H. Ishii (2010). "g-stalt: a chirocentric, spatiotemporal, and telekinetic gestural interface". In Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction (TEI '10). ACM, New York, NY, USA, 261-264. DOI=10.1145/1709886.1709939 http://doi.acm.org/10.1145/1709886.1709939

[13] D. Kirsh (1995) "The intelligent use of space". Department of cognitive science, UCSC, La Jolla, CA. Junio 1995, available at http://www.ida.liu.se/~729G12/mtrl/intelligent_use_of_space.pdf

[14] A. Kumpf (2009) "Trackmate: Large-Scale Accessibility of Tangible User Interfaces". Masters thesis. Masters of Science. Massachusetts Institute of Technology

[15] X. Xiao, M.S. Bernstein, L. Yao, D. Lakatos, L. Gust, K. Acquah, H. Ishii (2011). "PingPong++: community customization in games and entertainment". In Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology (ACE '11), Teresa Romão, Nuno Correia, Masahiko Inami, Hirokasu Kato, Rui Prada, Tsutomu Terada, Eduardo Dias, and Teresa Chambel (Eds.). ACM, New York, NY, USA, , Article 24 , 6 pages. DOI=10.1145/2071423.2071453 http://doi.acm.org/10.1145/2071423.2071453

[16] Marcosalonso (2012). "Reactable: basic demo #2". Visited january 2012. available at http://www.youtube.com/watch?v=MPG-LYoW27E

[17] J. Lee, R. Post, H. Ishii (2011). "ZeroN: mid-air tangible interaction enabled by computer controlled magnetic levitation". In Proceedings of the 24th annual ACM symposium on User interface software and technology (UIST '11). ACM, New York, NY, USA, 327-336. DOI=10.1145/2047196.2047239 http://doi.acm.org/10.1145/2047196.2047239

[18] J. Paradiso, C. Abler, K. Hsiao, M. Reynolds. (1997). "The magic carpet: physical sensing for immersive environments". CHI '97 Extended Abstracts on Human Factors in Computing Systems (CHI EA '97). ACM, New York, NY, USA, 277-278. DOI=10.1145/1120212.1120391 http://doi.acm.org/10.1145/1120212.1120391

[19] L. Chan, L. Hyk (2010.). "The magicpad: a spatial augmented reality based user interface". The 2010 Virtual Concept International Conferences (IDMME 2010), Bordeaux, France, 20-22 October 2010. In Proceedings of the IDMME, 2010.

[20] K. Sitdhisanguan, N. Chotikakamthorn, A. Dechaboon, and P. Out (2012). Using tangible user interfaces in computer-based training systems for low-functioning autistic children. Personal Ubiquitous Comput. 16, 2 (February 2012), 143-155. DOI=10.1007/s00779-011-0382-4 http://dx.doi.org/10.1007/s00779-011-0382-4

[21] P. Lapides, E. Sharlin, and M. Costa Sousa (2008). Three dimensional tangible user interface for controlling a robotic team. In Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction (HRI '08). ACM, New York, NY, USA, 343-350. DOI=10.1145/1349822.1349867 http://doi.acm.org/10.1145/1349822.1349867

[22] O. Shaer and R. J.K. Jacob (2009). A specification paradigm for the design and implementation of tangible user interfaces. ACM Trans. Comput.-Hum. Interact. 16, 4, Article 20 (November 2009), 39 pages. DOI=10.1145/1614390.1614395 http://doi.acm.org/10.1145/1614390.1614395

[23] O. Shaer, N. Leland, E. H. Calvillo-Gamez, and R. J. K. Jacob (2004). The TAC paradigm: specifying tangible user interfaces. Personal Ubiquitous Comput. 8, 5 (September 2004), 359-369. DOI=10.1007/s00779-004-0298-3 http://dx.doi.org/10.1007/s00779-004-0298-3

[24] A. Tang, M. Pahud, S. Carpendale, and B. Buxton (2010). VisTACO: visualizing tabletop collaboration. In ACM International Conference on Interactive Tabletops and Surfaces (ITS '10). ACM, New York, NY, USA, 29-38. DOI=10.1145/1936652.1936659 http://doi.acm.org/10.1145/1936652.1936659

[25] A. Kharrufa, R. Martinez-Maldonado, J. Kay, and P. Olivier (2013). Extending tabletop application design to the classroom. In Proceedings of the 2013 ACM international conference on Interactive tabletops and

surfaces (ITS '13). ACM, New York, NY, USA, 115-124. DOI=10.1145/2512349.2512816 http://doi.acm.org/10.1145/2512349.2512816

[26] A. Clayphan, A. Collins, C. Ackad, B. Kummerfeld, and J. Kay (2011). Firestorm: a brainstorming application for collaborative group work at tabletops. In Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '11). ACM, New York, NY, USA, 162-171. DOI=10.1145/2076354.2076386 http://doi.acm.org/10.1145/2076354.2076386

[27] M. Horn, Z. Atrash Leong, F. Block, J. Diamond, E. M. Evans, B. Phillips, and C. Shen (2012). Of BATs and APEs: an interactive tabletop game for natural history museums. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12). ACM, New York, NY, USA, 2059-2068. DOI=10.1145/2207676.2208355 http://doi.acm.org/10.1145/2207676.2208355

[28] A. Tabard, J. Hincapié-Ramos, M. Esbensen, and J. E. Bardram (2011). The eLabBench: an interactive tabletop system for the biology laboratory. In Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '11). ACM, New York, NY, USA, 202-211. DOI=10.1145/2076354.2076391 http://doi.acm.org/10.1145/2076354.2076391

[29] M. Khalilbeigi, J. Steimle, J. Riemann, N. Dezfuli, M. Mühlhäuser, and J. D. Hollan (2013). ObjecTop: occlusion awareness of physical objects on interactive tabletops. In Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces (ITS '13). ACM, New York, NY, USA, 255-264. DOI=10.1145/2512349.2512806 http://doi.acm.org/10.1145/2512349.2512806

[30] S. Klemmer. Katayanagi lectura at Carnegie mellon university. Video: http://www.youtube.com/watch?v=-QXEb6XSvXs

[31] A. Grosskopf, M. Weske, J. Edelman, M. Steinert, L. Leifer. Design Thinking implemented in Software Engineering Tools Proposing and Applying the Design Thinking Transformation Framework. hassoPlattner-Institute, University Potsdam, Germany, center for Design research, Stanford University, cA, USA. Disponible http://www.dab.uts.edu.au/research/conferences/dtrs8/docs/DTRS8-Luebbe-et-al.pdf

[32] T. Brown and B. Katz (2009) Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation, Harper Business, New York

[33] A. Fox and D. Patterson (2013). Engineering Software as a Service: An Agile Approach Using Cloud Computing. Strawberry Canyon LLC. ISBN-10: 0984881247.

AUTHORS' PROFILE

Christian C. Ventes. Software Engineer with Animation and Interactive Systems emphasis from the Pontificia Universidad Javeriana, Cali in Colombia. Interested in the areas of Human computer interaction, Interaction design, Interface development, user experience, affective computation and Computer graphics specialized in video games development.

Andrés A. Navarro-Newball. MSc and PhD in Computer Graphics. Associate Professor from the Electronics and Computer Science Department at the Pontificia Universidad Javeriana, Cali. WEB: http://cic.puj.edu.co/~anavarro

Deivy A. Velasco. Software Engineer from the Fundación Universitaria de Popayán University and master in computer science from the Pontificia Universidad javeriana cali in Colombia. Specialized in Interactive Systems, such as Augmented reality and speech processing.

Edmond C. Prakash. Professor in Computer Games Programming in the Department of Creative Technology, Faculty of Science and Technology, Bournemouth University, United Kingdom.