5* Magic Wand: a RGBD camera-based 5 DoF user interface for 3D interaction

Alexandre M. F. de Sousa
Institute of Mathematics and Statistics
University of São Paulo
São Paulo, Brazil
alemart@ime.usp.br

Carlos H. Morimoto
Institute of Mathematics and Statistics
University of São Paulo
São Paulo, Brazil
hitoshi@ime.usp.br

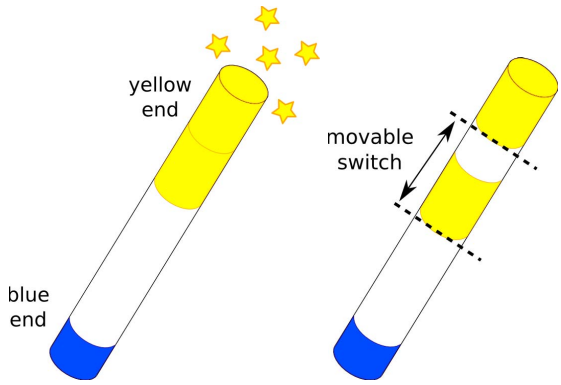
***Abstract*—This paper introduces the 5* Magic Wand: a user interface that provides an inexpensive way for input of 3D data. Drawing upon the metaphor of the "wizard", a user can point to places, perform gestures and cast "spells" to interact with the computer. We describe a computer vision technique that tracks the wand with 5 degrees of freedom. A prototype implementation, composed of a standard PC, a RGBD camera and a hand-made stick, is presented. Finally, a demo application showing how the wand may be used to navigate on planet Earth is presented.**

***Keywords*-3D user interface; magic wand; virtual reality; RGBD sensor; Kinect.**

I. Introduction

As technology drive us away from office spaces, interacting with computers like we do with the traditional desktop environment is no longer enough. In Human-Computer Interaction (HCI) research, advances in hardware and software have stimulated the development of a new generation of user interfaces called post-WIMP. Post-WIMP interfaces contain at least one interaction technique that is not based on classical 2D widgets such as menus and icons [1]. The motivation behind this new generation of interfaces is to employ the mundane, pre-existing knowledge of users as a leverage for reducing the gap between human goals and the actions required to accomplish them [2].

One important application of post-WIMP interfaces is the interaction in virtual environments (VEs). In particular, travel in virtual environments (VEs) is the most common interaction activity in 3D user interfaces [3]. Such interfaces should be easy to learn and easy to use, becoming "second nature" to users. As our interaction with computers expand beyond the standard desktop setting, it becomes desirable to explore different input settings. One possible approach is to use a wand. The affordances of that physical tool can be exploited in a variety of ways; it can be mapped to different interaction tasks in VEs. Using the notion of a magic wand for interaction makes it easy for the regular user to interact with the virtual world, as it is non-obstructive and removes the learning curve of less intuitive equipment such as the 3D mice [4]. In a recent study, Khan *et al.* [5] have shown that adopting a wand is an efficient manner to move within a VE, compared to relying on the position of the user in the physical space. Additionally, in comparison to using gloves or a 3D mouse, wand users tend to adopt more comfortable postures (they tend to find postures they like rather than what they feel is required) [6].



(a) *Active.* **(b)** *Inactive.*

Figure 1: *The 5* Magic Wand - two modes of operation.*

Wands also find application in diverse areas such as smart-home environments [7], education and the interactive arts. A magic wand could be used in a smart-home to turn a light on/off, control a television and so on. A teacher could use a magic wand in a classroom to point to and manipulate an object in 3D (e.g., a cell, a chemical molecule or a machine) in order to aid an explanation, if such a input device was cheap and easy to deploy. Similarly, for storytelling purposes, an artist could use a magic wand as part of an artistic performance, mixing both the real and the virtual worlds.

Nowadays, RGBD cameras are becoming inexpensive and widespread. These cameras provide color data as well as estimated depth for each pixel. In November 2010, Microsoft released the Kinect, a popular RGBD sensor that is an order of magnitude cheaper than earlier similar devices [8]. More recently, Intel has announced the RealSense[1], a RGBD sensor that may be embedded in everyday devices such as laptops in the near future. RGBD cameras have a wide range of applications, including: interaction, augmented reality, 3D reconstruction and robotics [9]. The potential ubiquity of such sensors in the future make them an attractive choice for implementing an interface such as a magic wand for 3D interaction.

[1]https://software.intel.com/en-us/realsense/. Last access: March 11th, 2015.

978-1-4673-7204-6/15 $31.00 © 2015 IEEE
DOI 10.1109/SVR.2015.10

This paper presents the 5* Magic Wand: a minimalist 5 DoF 3D user interface that can be built inexpensively and that draws upon users' knowledge carried over from childhood (e.g., fantasy movies). Unlike similar works, the 5* Magic Wand includes a on/off switch to enhance interaction capabilities and is tracked using one RGBD camera, making it resilient against background noise. A novel computer vision technique to track the wand is described. As an application example, we combine pointing, gestures and speech to show how the wand can be used to navigate on planet Earth.

The rest of this paper is organized as follows. Section II presents related work. Section III discusses the design of the 5* Magic Wand. Section IV discusses the system implementation, the computer vision algorithms to track the wand with five degrees of freedom and presents a working prototype. Section V presents a demo application and section VI concludes the paper.

II. Related Work

Different kinds of wands have been used for travelling VEs. Using magnetic technology, Ciger *et al.* [4] have presented "The Magic Wand", a user interface for interaction with a VE displayed on a large projection screen. The user flies on a magic "carpet", exploring the landscape. The wand is used to point to regions of the scene, and a small vocabulary of voice commands ("spells") directs the simulator. The system recognizes four basic postures/orientations of the wand. According to the authors, unlike the orientation data, the position data returned by the tracker is very innacurate (and thus is discarded).

Using a pair of calibrated stereo cameras, Guo *et al.* [10] have designed the "Featured Wand", a passive wand with two colored end markers and a spiral marking in between. A computer vision system tracks the position and the orientation of the wand in 3D space, at 9 frames per second. The authors have subsequently used the wand for navigation in VEs: tilting up or down controls the inclination of the viewpoint, and rotating left or right controls the heading. Raising or lowering control elevation, and lateral movements change the panning.

Cabral *et al.* [11] have developed a collaborative application that works in a CAVE system equipped with a Optitrack System featuring 4 infrared cameras. One of the users of the application, called the explorer, uses a Wiimote coupled with a reflective marker to point where he wants to navigate to (the Wiimote is tracked for its position and orientation). Then, he pushes a button to pull himself to that place. That interaction metaphor, called "point and go", was used in a closed VE (i.e., no sky).

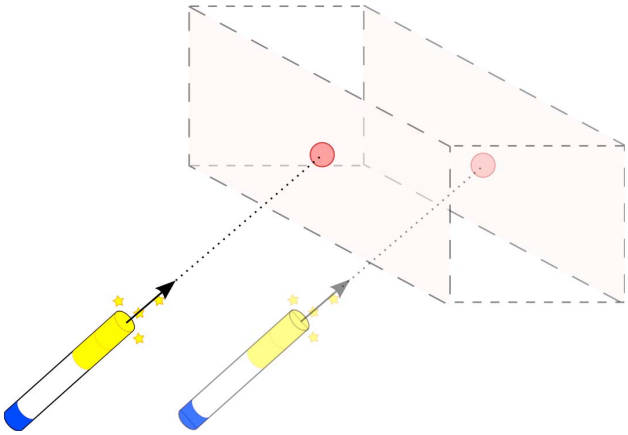


Figure 2: *The wand as a 3D pointing device.*

III. Design

The design goals of the 5* Magic Wand can be ennumerated as follows:

1) it should be tracked for its position and orientation in 3D space;
2) it should enable users to point to elements on a screen;
3) it should present a mechanism for enabling the execution of "spells" (gestures), hence giving the wand its magic touch;
4) it should be built easily and inexpensively;
5) it should be non-obstructive.

The proposed design for our wand brings features that enable its tracking by a RGBD camera. We start with a plastic rod with length of about $30cm$. Colored markings at each end of the wand (see Fig. 1a) enable its tracking by a computer vision system. This particular configuration allows 5 DoF pose estimation (position in 3D space plus two rotation angles: yaw and pitch). One can then define a 3D ray starting from the head of the wand pointing to the direction of the rod. Once the coordinates of the screen are known, it's easy to use the wand as a pointing device (see Fig. 2).

In addition to the above, we have designed a switch: a movable part near one of the extremes of the wand will indicate whenever a "spell" begins and whenever it ends (see Fig. 1b). The switch may be moved using a thumb, and it works like a on/off element. Whenever it overlaps the corresponding extreme of the device (i.e., the extreme sharing the same color), the wand is said to be active. Otherwise, it's said to be inactive.

The design of our passive wand (i.e., no electronic components, no wires, etc.) enables it to be built easily and inexpensively and makes it a non-obstructive interaction device.

IV. Implementation

As shown in Figure 4, the computational implementation of the 5* Magic Wand is divided in three layers:

- **Sensory layer:** a RGBD camera will capture color and depth data from the environment;
- **Tracking layer:** this layer determines the position, orientation and state (active or inactive) of the wand;
- **Application layer:** uses the wand as an input device for some activity (e.g., travel in a VE).

Regarding the sensory layer, a Kinect camera is used. Its availability and affordable cost make it an attractive choice for implementing the wand. The tracking layer brings technical challenges that will be addressed in more detail.

A. Tracking layer

As depicted in Figure 3, given the RGBD data from the sensory layer, a sequence of steps is employed to extract the position, orientation and state (active or inactive) of the wand.

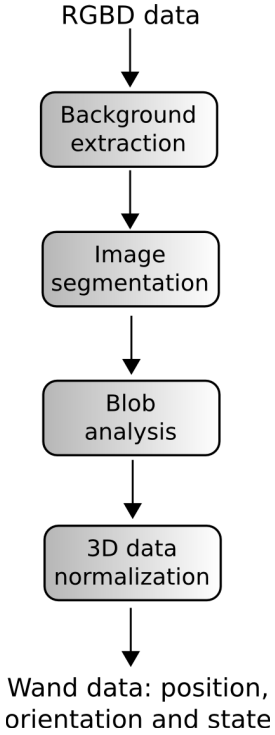


Figure 3: *Steps performed by the wand tracker.*

1) Background extraction: Unlike similar works based on optical tracking, tracking the wand with a RGBD camera make it resilient against background noise, since the background can be extracted using depth data. Additionally, past research has indicated that the depth estimates provided by the Kinect are quite stable [12].

A background model is created by taking n snapshots $\{S_1, S_2, \ldots, S_n\}$ of the depth image. Denoting the depth value of the i-th pixel of S_j by d_{ij}, let μ_j and σ_j be, respectively, the mean and the standard deviation of $\{d_{1j}, \ldots, d_{nj}\}$. The background model is said to be the collection of all μ_j. The set of all σ_j is used as a noise profile.

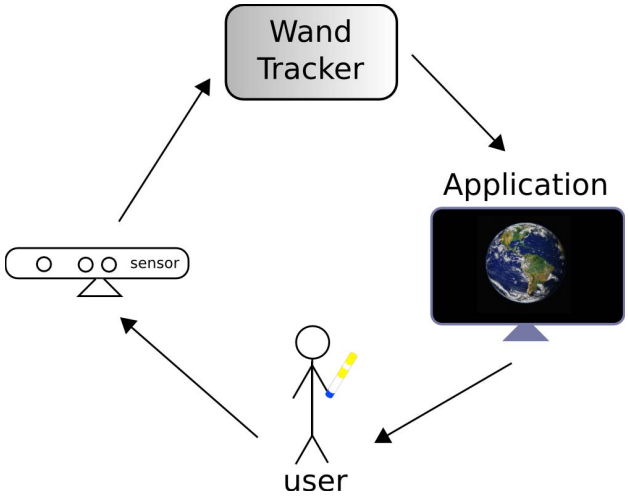


Figure 4: *System overview.*

The background extraction is performed by subtracting the newly acquired depth images from the background model.

2) Image segmentation: After the previous step, the color image provided by the RGBD sensor features artifacts belonging to the foreground. The colored markings of the wand are segmented by converting the color image to the HSV space and employing a thresholding method.

3) Blob analysis: The resulting blobs, computed using connected components analysis, indicate the colored markings. In addition to the blue marking, the wand features two yellow markings: one corresponds to the on/off switch and the other indicates one of the ends of the wand. If the switch is on, the markings overlap, meaning that only one yellow blob will be visible in the color image. Therefore, the state of the wand is set to be "inactive". If there are two yellow blobs, the wand is "active".

4) 3D data normalization: A RGBD sensor can provide the 3D position, in its own coordinate system, of any pixel [13]. That said, the yellow markings are used to determine the position of the wand in 3D space. The position is taken to be 3D location of the center of mass of the yellow blob (or the mean of the centers of mass if there is more than one yellow blob, i.e., if the wand is active).

In order for the wand to be usable to the application layer, one needs to normalize its position and orientation to account for different placements of the camera. Let R be the cubic region described by:

$$R : \lim_{t\to\infty} \sqrt[t]{\left|x-\frac{1}{2}\right|^t + \left|y-\frac{1}{2}\right|^t + \left|z-\frac{1}{2}\right|^t} \leq \frac{1}{2}$$

We create a transformation T that maps a position from the camera coordinate system to the normalized space described by R. Let (x_k, y_k, z_k) be the coordinates of the wand in the camera system and (x, y, z) be the corresponding normalized position. Given model parameters $(a, b, c, d, e, f, g, h, i, j, k, l)$, mapping T is defined as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ z_k \\ 1 \end{bmatrix}$$

A calibration procedure is employed to estimate the model parameters. Given a set of $n \geq 4$ correspondences, denoted by $(x^{(j)}, y^{(j)}, z^{(j)}) \leftrightarrow ({x_k}^{(j)}, {y_k}^{(j)}, {z_k}^{(j)})$ for $1 \leq j \leq n$, the above equation can be rewritten to:

$$\begin{bmatrix} x^{(1)} \\ y^{(1)} \\ z^{(1)} \\ x^{(2)} \\ y^{(2)} \\ z^{(2)} \\ \vdots \\ x^{(n)} \\ y^{(n)} \\ z^{(n)} \end{bmatrix} = \begin{bmatrix} {x_k}^{(1)} & {y_k}^{(1)} & {z_k}^{(1)} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & {x_k}^{(1)} & {y_k}^{(1)} & {z_k}^{(1)} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & {x_k}^{(1)} & {y_k}^{(1)} & {z_k}^{(1)} & 1 \\ {x_k}^{(2)} & {y_k}^{(2)} & {z_k}^{(2)} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & {x_k}^{(2)} & {y_k}^{(2)} & {z_k}^{(2)} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & {x_k}^{(2)} & {y_k}^{(2)} & {z_k}^{(2)} & 1 \\ & & & & & & \ddots & & & & & \\ {x_k}^{(n)} & {y_k}^{(n)} & {z_k}^{(n)} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & {x_k}^{(n)} & {y_k}^{(n)} & {z_k}^{(n)} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & {x_k}^{(n)} & {y_k}^{(n)} & {z_k}^{(n)} & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \\ j \\ k \\ l \end{bmatrix}$$

A least-squares solution for the model parameters can be found by applying a pseudoinverse matrix, computed using singular value decomposition, QR factorization or other methods [14].

From the end-user point of view, one does not need to manually describe the correspondences. The calibration procedure simply requires that the user moves the wand near the camera (see Fig. 5). Let the curve $\gamma(t) : (x_k(t), y_k(t), z_k(t))$ describe such a movement, where $t \geq 0$ denotes the time factor. We proceed by defining what we call box functions:

$${x_k}^{\#}(d) = d \cdot \max_{t \geq 0} x_k(t) \; + \; (1-d) \cdot \min_{t \geq 0} x_k(t)$$

$${y_k}^{\#}(d) = d \cdot \max_{t \geq 0} y_k(t) \; + \; (1-d) \cdot \min_{t \geq 0} y_k(t)$$

$${z_k}^{\#}(d) = (1-d) \cdot \max_{t \geq 0} z_k(t) \; + \; d \cdot \min_{t \geq 0} z_k(t)$$

For $i \in \mathbb{N}^*$, let us define the family of functions:

$$J_i(j) = \left\lfloor \frac{j-1}{2^{(i-1)}} \right\rfloor - 2 \left\lfloor \frac{j-1}{2^i} \right\rfloor$$

Then, for $j = 1, 2, \ldots 8$, we set the correspondences:

$$x^{(j)} = J_1(j) \; \leftrightarrow \; {x_k}^{(j)} = {x_k}^{\#}(J_1(j))$$

$$y^{(j)} = J_2(j) \; \leftrightarrow \; {y_k}^{(j)} = {y_k}^{\#}(J_2(j))$$

$$z^{(j)} = J_3(j) \; \leftrightarrow \; {z_k}^{(j)} = {z_k}^{\#}(J_3(j))$$

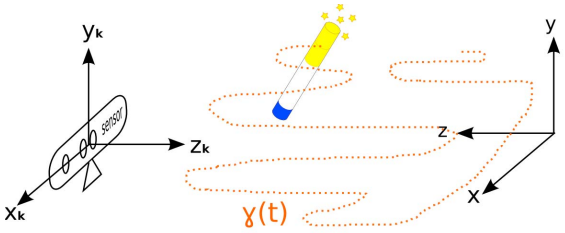


Figure 5: *Calibration procedure.*

Once the model parameters have been estimated, the 3D position in the normalized space of both ends of the wand will determine its orientation. This phase ends with a state (active or inactive) and a pose estimation with five degrees of freedom: the 3D position of the wand plus two rotation angles (yaw and pitch).

B. Application layer

The tracking layer communicates wand data to the application layer. As displayed in Figure 6, an HTML5 application has access to the position, orientation and state of the wand. Additionally, a gesture recognition module enhances the interactive capabilities of our wand, turning it into a magic tool.

According to Mitra and Acharya, gestures are meaningful body movements involving fingers, arms, head or the full body in order to convey information or interact with the environment [15]. Ghirotti and Morimoto have pointed out two main reasons for using gestures for interaction with computers [16]:

1) People use a wide range of gestures in everyday life. New gestures can be easily and quickly learnt by observing others perform them;

2) Gesture-based interfaces allow the "natural" usage of "gestural phrases". These phrases can be thought of as a way to break the communication dialog in parts. These parts have simple meanings and can be easily interpreted by computers (e.g., the action of moving an object in a VE can be decomposed in three parts: grab, translate and release it).

Gestures can be static (the person assumes a specific pose), dynamic (there is ongoing movement) or a mixture of the two (used in sign languages). Automatic recognition of dynamic gestures requires temporal segmentation, and often the user has to specify the beginning and the ending points of a gesture in space and time [15].

The 5* Magic Wand includes an on/off switch. Whenever the user pulls the switch away from the nearest ending of the wand (i.e., whenever the wand becomes active), we'll say that a spell has begun. Whenever the user pushes the switch back to its previous position (i.e., the wand becomes inactive), we'll say that a spell has ended. In

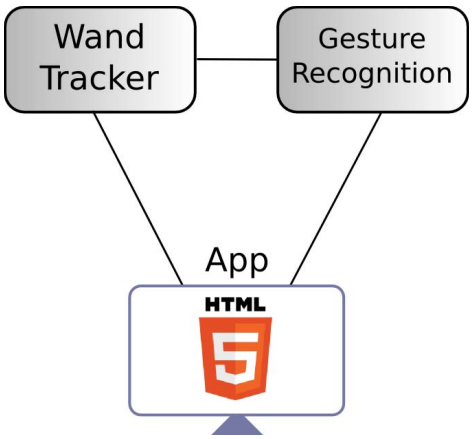


Figure 6: *The wand can control an HTML5 application.*

this work, a spell is defined to be the trajectory performed by the wand when its state is active.

The Gesture Recognition block depicted in Figure 6 receives a spell from the Wand Tracker and assigns it a label (e.g., "circle", "arrow", "hat" and so forth). The advent of lightweight and robust gesture recognizers has enabled rapid prototyping of gesture-based user interfaces [17], [18]. After normalizing the gestures, often adding position, scale and rotation invariance, they employ template matching techniques for recognition. These recognizers usually rely on simple geometry and trigonometry and perform remarkably well with only a few training examples, making them an attractive choice for enabling users to cast spells with our wand. Figure 7 shows a prototype implementation running at about 28 frames per second on a Dell All-in-One Inspiron 2330 PC (with 6 GB RAM and an Intel Core i5 CPU at 2.30 GHz).

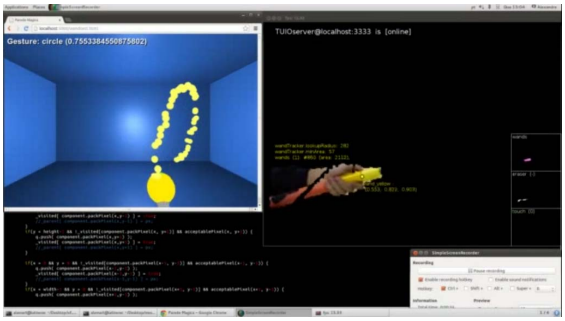
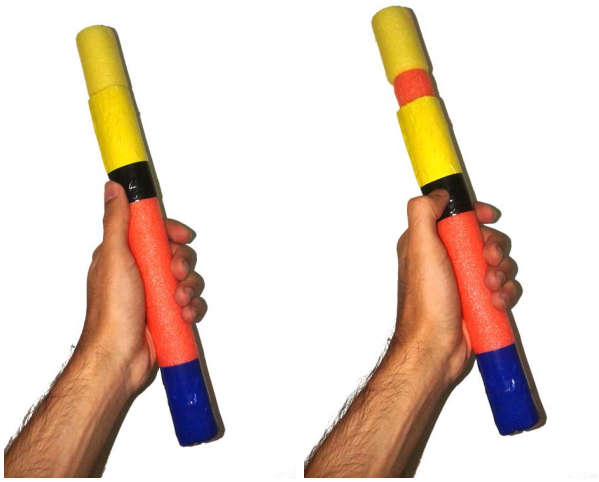


Figure 7: *Casting spells: a prototype implementation.*

V. Demo Application

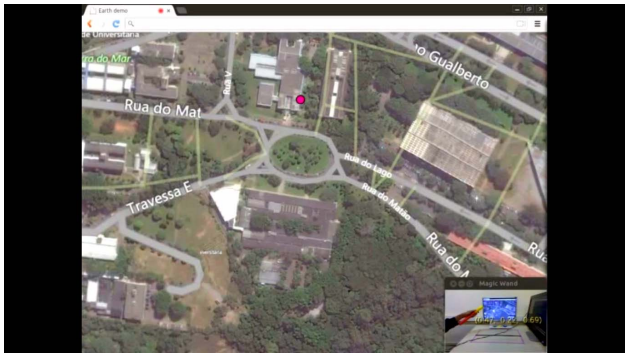
We have applied wand based input for a 3D navigation activity. The desirable features for the simulator are:

1) The Earth may be viewed in multiple scales, ranging from a full-planet to a street-level view;
2) The user may point to places (such as when giving a lecture);



(a) *The minimalist nature of the magic wand.*

(b) *Planet Earth.*



(c) *Map-like view.*

(d) *Just changed to continuous mode.*

Figure 8: *The demo.*

3) The user may "bookmark" places (i.e., instantly remove, record and jump to previously recorded locations). We'll call these places checkpoints;
4) The user may fly over any place on the planet.

In this work, we combine wand input and speech to demonstrate how one can use the wand to navigate on planet Earth. Multimodal interfaces can support efficient and expressive means of human-computer interaction that are more akin to how humans experience the physical world [19]. Past works have also employed a multimodal approach for navigation in virtual environments [4], [20].

The Wand Tracker, built in C++, communicates wand data to the HTML5 application through the TUIO protocol [21] over WebSockets. The Earth demo is built using Cesium, an open source WebGL Virtual Globe and Map Engine [2]. Speech recognition is performed using the HTML5 Web Speech API.

The simulator features two navigation modes:

- **Discrete mode**: the wand (see Fig. 8a) is used mainly as a pointing device. The user may issue commands using speech and views the planet in a map-like fashion (see Figs. 8b, 8c). Navigation occurs in a discrete, command-like manner;
- **Continuous mode**: wand input is mapped to continuous viewpoint movement. The end result somewhat resembles a user flying around in a helicopter (see Fig. 8d). Boomarking commands may also be issued using speech.

Two spells are supported by the simulator: circle and arrow. Whenever the user performs the circle spell, the discrete mode is activated. The arrow spell activates the continuous mode, making the camera look towards the direction pointed by the arrow (see Figs. 8d, 9a).

A. Discrete mode

The wand acts as a pointing device. The user may point to places on the screen and issue commands using speech. Table I summarizes the supported commands.

Command	Resulting action
Centralize	Centralizes the viewer on the spot pointed by the wand.
Move closer	Zooms in the viewer on the spot pointed by the wand.
Move away	Zooms out.
Here's <place>	Creates a checkpoint named <place> on the spot pointed by the wand (see Fig. 9a).
Remove <place>	Removes an existing checkpoint named <place>.
Go to <place>	Makes the simulator move the camera to the checkpoint named <place>.

Table I: *Supported commands in discrete mode.*

Besides pointing to places and managing the bookmarks, navigation on multiple scales can be performed by zooming the map in or out.

[2]https://cesiumjs.org/. Last access: March 15th, 2015.

(a) *Creating a checkpoint and changing the direction of the view.*

(b) *Moving forwards.*

(c) *Moving upwards.*

(d) *High altitude.*

Figure 9: *Flying around in continuous mode.*

B. Continuous mode

In continuous mode, wand input is mapped directly to viewpoint movement. Whenever the wand is close to the screen, the viewpoint moves forward (see Fig. 9b).

Positioning the wand far away from the screen makes the viewpoint move backwards. Moving the wand left or right controls panning. Raising or lowering it controls the altitude (see Figs. 9c, 9d).

Unlike the discrete mode, in which a map-like view of the planet is displayed, the continuous mode suggests that the user is flying around the Earth.

An important feature to be considered when designing user interfaces is the rest pose. The rest pose is important to establish a situation where there is no interaction with the system, so that the user may not be tired after a short period of interaction [16]. In order to rest, the user may instruct the system to stop to flight (which is enabled by default) or simply switch to the discrete mode of navigation. Table II presents the supported commands.

Command	Resulting action
Stop flight	The wand input will no longer be mapped to viewpoint movements.
Start flight	After stopping the flight, this command makes the wand input be mapped to viewpoint movements once again.
Here's <place>	Creates a checkpoint named <place>.
Remove <place>	Removes checkpoint <place>.
Go to <place>	Moves the camera to checkpoint <place>.

Table II: *Supported commands in continuous mode.*

VI. Conclusion

This paper has presented the 5* Magic Wand: a user interface for 3D input that provides position and orientation. Its physical shape makes it suitable for pointing, gesturing and casting magical "spells". The presence of the on/off switch expands the capabilities of the device. The simple hardware setup of the wand eliminates the need of wires/electronic components, hence it is unobstructive and can be built quickly and inexpensively. A computer vision technique to track the wand in 3D space with a RGBD camera was presented. Finally, it was shown how the wand may be used to navigate in a virtual environment.

Given the potential high availability of RGBD cameras in the near future (e.g., embedded in everyday computers) and the minimalist nature of the 5* Magic Wand, we believe that this work may help bring 3D wand interaction to everyday settings such as education.

Possible improvements of this work include: explore interaction possibilities when one points the bottom (blue) part of the wand towards the screen, include a 6th degree of freedom (twist) and use the push-pull mechanism of the switch as a source of continuous data rather than a on/off one. Additionally, the passive and inexpensive nature of the wand enables it to be used for controlling games.

References

[1] A. Van Dam, "Post-wimp user interfaces," *Communications of the ACM*, vol. 40, no. 2, pp. 63–67, 1997.

[2] R. J. Jacob, A. Girouard, L. M. Hirshfield, M. S. Horn, O. Shaer, E. T. Solovey, and J. Zigelbaum, "Reality-based interaction: a framework for post-wimp interfaces," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2008, pp. 201–210.

[3] D. A. Bowman, E. Kruijff, J. J. LaViola Jr, and I. Poupyrev, *3D user interfaces: theory and practice*. Addison-Wesley, 2004.

[4] J. Ciger, M. Gutierrez, F. Vexo, and D. Thalmann, "The magic wand," in *Proceedings of the 19th spring conference on Computer graphics*. ACM, 2003, pp. 119–124.

[5] V. Khan, M. Pekelharing, and N. Desle, "Efficient navigation in virtual environments: A comparative study of two interaction techniques: The magic wand vs. the human joystick," in *Intelligent Human Computer Interaction (IHCI), 2012 4th International Conference on*. IEEE, 2012, pp. 1–5.

[6] M. Henschke, T. Gedeon, R. Jones, S. Caldwell, and D. Zhu, "Wands are magic: a comparison of devices used in 3d pointing interfaces," in *Human-Computer Interaction–INTERACT 2013*. Springer, 2013, pp. 512–519.

[7] A. Wilson and S. Shafer, "Xwand: Ui for intelligent spaces," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2003, pp. 545–552.

[8] K. Litomisky, "Consumer rgb-d cameras and their applications," *University of California, Riverside. Ano*, 2012.

[9] L. Cruz, D. Lucio, and L. Velho, "Kinect and rgbd images: Challenges and applications," in *Graphics, Patterns and Images Tutorials (SIBGRAPI-T), 2012 25th SIBGRAPI Conference on*. IEEE, 2012, pp. 36–49.

[10] F. Guo, D. Kimber, and E. G. Rieffel, "Featured wand for 3d interaction." in *ICME*, 2007, pp. 2230–2233.

[11] M. Cabral, G. Roque, D. dos Santos, L. Paulucci, and M. Zuffo, "Point and go: Exploring 3d virtual environments," in *3D User Interfaces (3DUI), 2012 IEEE Symposium on*. IEEE, 2012, pp. 183–184.

[12] A. D. Wilson, "Using a depth camera as a touch sensor," in *ACM international conference on interactive tabletops and surfaces*. ACM, 2010, pp. 69–72.

[13] J. Kramer, M. Parker, D. Herrera, N. Burrus, and F. Echtler, *Hacking the Kinect*. Apress, 2012.

[14] L. N. Trefethen and D. Bau III, *Numerical linear algebra*. Siam, 1997, vol. 50.

[15] S. Mitra and T. Acharya, "Gesture recognition: A survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, no. 3, pp. 311–324, 2007.

[16] S. E. Ghirotti and C. H. Morimoto, "Um sistema de interação baseado em gestos manuais tridimensionais para ambientes virtuais," in *Proceedings of the IX Symposium on Human Factors in Computing Systems*. Brazilian Computer Society, 2010, pp. 159–168.

[17] J. O. Wobbrock, A. D. Wilson, and Y. Li, "Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes," in *Proceedings of the 20th annual ACM symposium on User interface software and technology*. ACM, 2007, pp. 159–168.

[18] Y. Li, "Protractor: a fast and accurate gesture recognizer," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2010, pp. 2169–2172.

[19] Y. Rogers, H. Sharp, and J. Preece, *Interaction design: beyond human-computer interaction*. John Wiley & Sons, 2007.

[20] D. M. Krum, O. Omoteso, W. Ribarsky, T. Starner, and L. F. Hodges, "Speech and gesture multimodal control of a whole earth 3 d visualization environment," in *ACM International Conference Proceeding Series*, vol. 22, 2002, pp. 195–200.

[21] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza, "Tuio: A protocol for table-top tangible user interfaces," in *Proc. of the The 6th Intl Workshop on Gesture in Human-Computer Interaction and Simulation*, 2005, pp. 1–5.