Medianeum: Gesture-Based Ergonomic Interaction

François Zajéga[1], Cécile Picard-Limpens[1], Julie René[2], Antonin Puleo[1], Justine Decuypere[1], Christian Frisson[1], Thierry Ravet[1], and Matei Mancaş[1]

[1] Numediart Institute, University of Mons, Bd. Dolez 31, Mons, Belgium
francois.zajega@umons.ac.be
[2] ISEN, Ecole d'Ingénieurs de Lille, France

Abstract. The proposed MEDIANEUM system consists in an interactive installation allowing general audiences to explore a timeline and access informational multimedia data such as texts, images and video.

Through a Microsoft Kinect depth sensor, users' skeletons are captured and their gestures are tracked to interact with the data presented on a screen in an ergonomic way.

The graphical user interface is built upon *ProcesSwing*, our version of the *Processing* IDE embedded into a standard Swing Java GUI widget toolkit application, and the TimelineJS library from Vérité.co/Northwestern University, allowing to create online, personalized and interactive timelines that mash up historical events, sorted in definable categories.

Keywords: timeline, Kinect, ergonomics, gestures, interface, interaction.

1 Introduction

The Medianeum project is an attempt to use gesture-based interfaces in museums. The prototype described in this paper is used at the Mundaneum [1], the archive centre of the French Community of Wallonia-Brussels and Temporary Exhibition Space. The application of the proposed gesture interface is the interaction of visitors with a timeline containing the life of Henri La Fontaine, one of the founder of the *Mundaneum*, and the related historical events. The proposed interface is able to handle the general audience which visit a museum. People use the interface to interact with the timeline in order to explore the different historical events presented/supported as video, images and texts and shown on a screen.

In section 2 we describe the proposed captured system. Section 3 describes the novel approach introduced in this paper for an optimized interaction. Sections 4 and 5 detail the use of events by the system and the graphic of the interface providing feedback tu the user. Finally we conclude in section 6.

[1] Mundaneum: http://www.mundaneum.org

M. Mancas et al. (Eds.): INTETAIN 2013, LNICST 124, pp. 96–103, 2013.

2 Capture System

To accelerate development , *Processing*[2] [2] has been chosen as a basis. It proposes a wide variety of libraries that are ready to use and easy to deploy. However, *Processing* was too limited to develop the whole application: we missed GUI elements such as sliders or buttons, as well as the possibility to integrate an HTML rendering engine. The core of *Processing* has been embedded into a standard Swing Java GUI widget toolkit application. We named this swing compliant version of *Processing ProcesSwing* [3]. It is available for download on the numediart website[3].

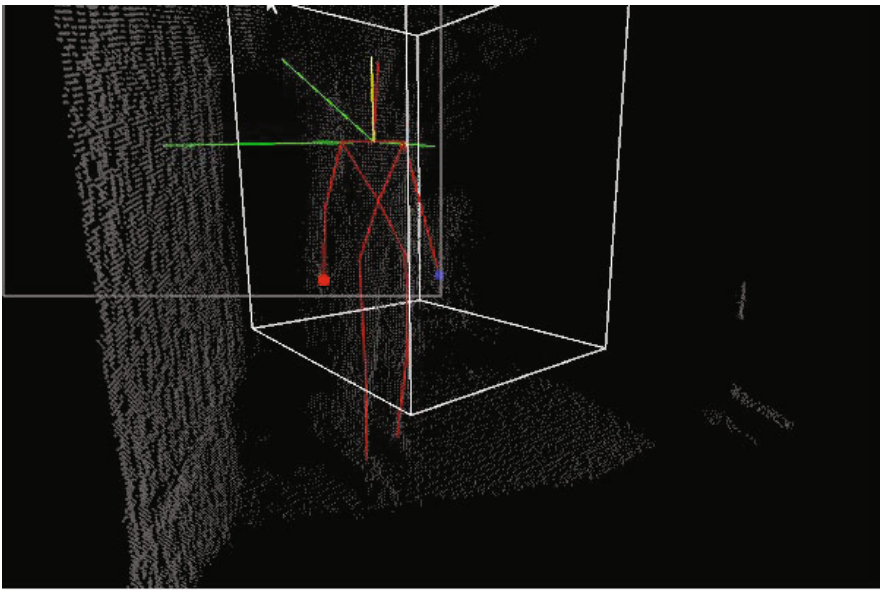

Fig. 1. We define an active area where the user can freely move its upper-body. The active area is represented by a parallelepiped in the virtual world.

Medianeum uses the Microsoft Kinect sensor which provides hands-free control capacities. For this purpose we use NITE, the open source drivers along with motion tracking middleware from PRIMESENSE, in association with OpenNI [4] [1] which allows us to retrieve the skeleton of users (in red in Figure 1).

We define *active* areas where the users must be located to be able to interact with the system. The area is defined by a parallelepiped in the virtual world and is marked as a dot on the ground or a ray of light in the physical world (see Figure 1). When the user, represented as its skeleton in the virtual world,

[2] Processing: `http://processing.org`
[3] numediart tools: `http://www.numediart.org/tools`
[4] OpenNI: `http://openni.org`

penetrates this parallelepiped, the system checks that both shoulders are in the active area. As soon as this condition is fulfilled, the user is considered as active.

3 Interaction Design

3.1 Early Approach: Planar Representation of the Gesture Movements

Most of the existing software providing similar interaction as KinVi [4] use a planar representation of gestures (Figure 2).

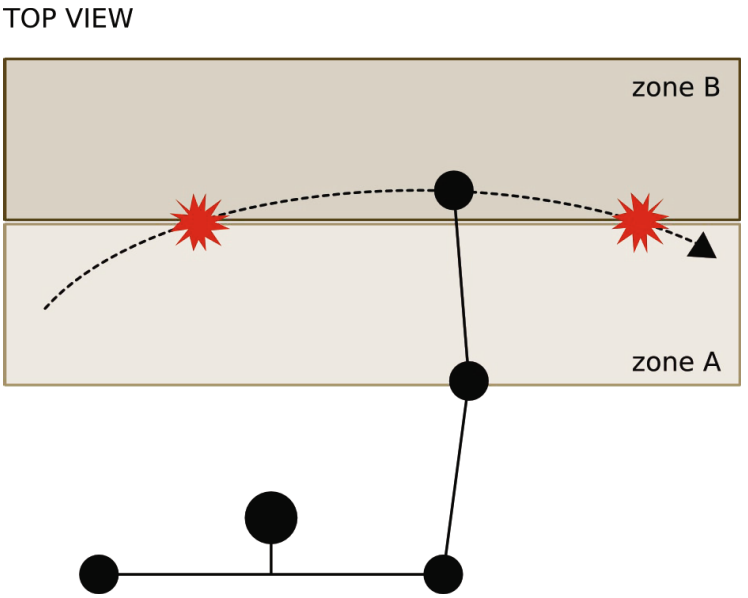

Fig. 2. In the KinVi approach, the relative depth of the hand is varying during displacement in the layout

However, when we try to imitate Kinvi-like interactions (Figure 3) we rapidly realize the limits of such an approach. In this case, a pop-up window showing the boxes and hands' positions in those boxes must be displayed for the user to understand the setup (Figure 2). In particular, the method has two main drawbacks:

1. Size of the pop-up window: to be understandable, the pop-up screen has to be relatively big. As a consequence, the screen area available to the content exploration is reduced.

   In addition, this pop-up window has been shown to distract the user from the important information.

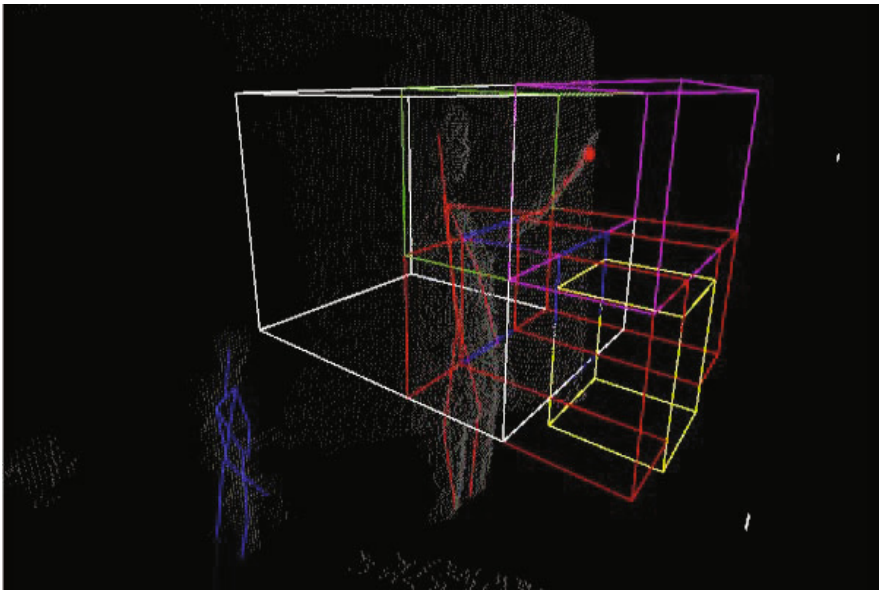

Fig. 3. First tests with a similar approach than KinVi interaction design. We place a big virtual parallelepiped, called *layout*, in front of the active area and divide it in two layers. The first layer is equivalent to a *mouse over* or *highlight* in a standard GUI environment. The second layer allows the user to click. Buttons are small parallelepiped, some having the two stages, some having only the highlight stage.

2. Ergonomics of human-system interaction: buttons are difficult to locate and very sensitive to shaking. In this case, a large layout is easier to use (wider areas) but forces the user to bend in order to reach the bottom buttons.During the displacement in the layout, the relative depth of the hand is varying (see Figure 2). This is not compatible with the parallelepiped shape of the buttons. To avoid that issue, we have to deepen all the buttons. This is, once again, inducing wider movements to reach the click area which can be painful in long interactions.

3.2 Final Approach: Spherical Representation of the Gesture Movements

In this case we developed a method which is intuitive enough to avoid displaying any pop-up window with user avatar and virtual controllers. The hand is simply represented as a pointer on the screen, leaving a larger space to the content to be explored.

However, this is not solving the ergonomic problem of parallelepiped shapes. A 2D dimensional projection of the hand on a virtual plane involves losing the possibility to use the depth of the pointer, and we want to keep this opportunity.

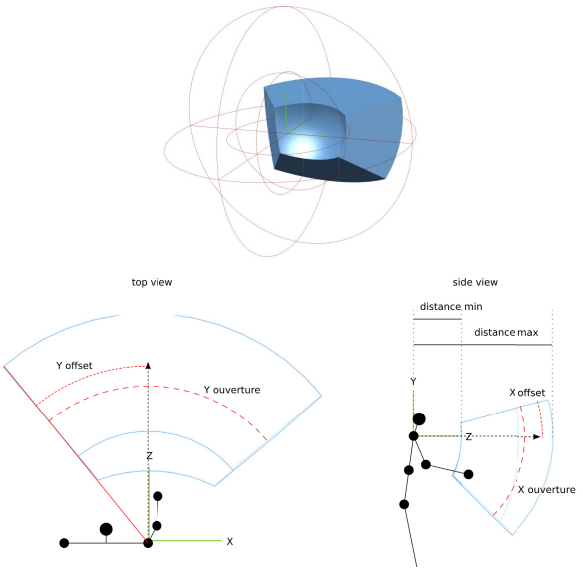

Fig. 4. Left: the parallelepiped layout that represents the active area is twisted to turn it into a portion of a sphere. Right: the position of the hand is calculated as a spherical position and the layout size is defined by four angles and two length.

We observe that when a user moves his arm from left to right, he doesn't follow straight lines. In our case, the user is not placed close to a table, a desk nor a wall. Thus, having no direct physical contact with a planar surface, the arm is moved following the natural morphology of the body. Based on this observation, we twist the virtual parallelepiped, called *layout*, to turn it into a portion of a sphere (see Figure 4, Left).

We attach the sphere to the shoulder linked to the active hand. The position of the hand is now calculated in spherical coordinates and no more in Cartesian ones.

X and Y-coordinates of the hand pointer are processed from the angles of the hand/shoulder vector in the XZ-plan and YZ-plan respectively. The Z-coordinate is calculated as the distance between the hand and the shoulder compared to the minimum and maximum sphere radius. The comparison with minimal and maximal values allows us to normalize all the axis between 0 and 1. This way of retrieving a 3 dimensional position increases theoretically the precision of the movement (see Figure 5). A planar projection tends to reduce the amplitude when the movement approaches the line perpendicular to the plane.

This layout is user-centric. By gluing the centre of the sphere to the shoulder, the interaction area stays at the right location, even if the user moves. The sphere is adapted according to the morphology of the user. The radius of the sphere is calculated proportionally to the length of the torso/neck distance, these two points being the most stable in the skeleton.

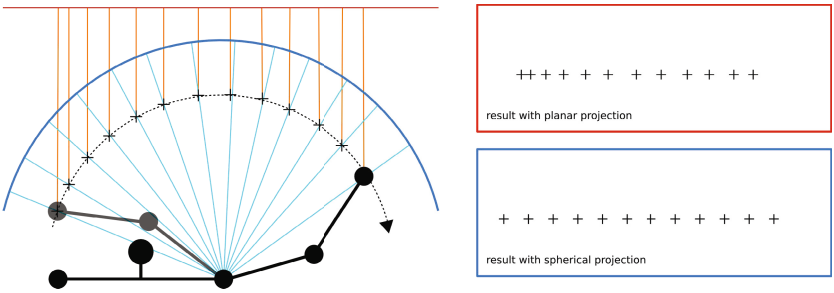

Fig. 5. A planar projection tends to reduce the gesture amplitude when the movement goes further from the barycentre of the skeleton, which is not the case for a spherical projection

In this configuration, we are able to detect click-like movements when the distance between the hand and the shoulder is quickly increasing.

4 Events Management

Skeleton movements of the active user is analysed by the system for appropriate feedback. We define events for interaction and two main categories of events are fired by the system.

1. *State changed* events
   - **User entering the active area.** This event is triggered when the two shoulders of the user are inside the parallelepiped defined as the the active area. If any is set as active yet, the user is flagged as active, preventing any other skeleton to be considered as active.
   - **User leaving the active area.** This event is triggered when the two shoulders of the user are outside the parallelepiped defined as the the active area. If the leaving user was active, the system will accept any other user in the active area.
2. *Interaction* events (the interaction events are obviously emitted when a user is set active.)
   - **Active user's hand entering the layout for the first time.** A specific event is triggered when the hand of the user enters the layout for the first time.
   - **Active user's hand inside the layout.** At each iteration,the position of the hand's user is broadcasted if if the hand is inside the layout.
   - **Active user's hand leaving the layout.** A specific event is triggered when the hand of the user leaves the layout.

5 Graphical Interface

The graphical interface is built on web technologies. Chromium is used as rendering engine and is embedded as a viewport using the `http://www.eclipse.org/swt/`SWT library[5] [5]. This allows us flexibility: since HTML is not compiled, it can be modified via a simple text editor without influencing the core of the application.

We use TimelineJS[6] library from Vérité.co/Northwestern University to display multimedia content. TimelineJS also proposes several ways to load the content: via the Google Doc API, via plain HTML, or via JavaScript Object Notation (JSON). TimelineJS is developed to be used with a mouse-like device. Due to the relatively poor precision of the pointer compared to a standard mouse, we implement a upper layer for control. All the buttons are set bigger and the drag method that allows vertical scrolling in the timeline is made easier (see Figure 6). To achieve this, a set of Javascript methods detect collisions with invisible areas placed above the interface. The active areas are also managed via Javascript to track interaction with the different elements.

Fig. 6. The graphical interface showing the timeline and the controllers for content navigation

Before launching the timeline, a series of screens welcome the user, allow him to select his language and give him a very short tutorial (see Figure 7). A screensaver is shown when no user is in the active area.

[5] `http://www.eclipse.org/swt/`
[6] `http://timeline.verite.co`

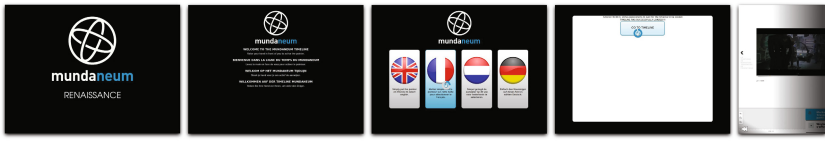

Fig. 7. The graphical interface showing the timeline and the controllers for content navigation

6 Conclusions

We built gesture-based system for complex scenarios such as museums where the users are heterogeneous which is:

1. Reliable from the capture point of view:
   An open space, unmanaged, is typically the worst place to install a precise motion detection.
2. Intuitive in terms of interface:
   This exhibition will be accessible to a broad audience and we can not suppose that the audience is already trained to computer interfaces similar to multiple touch screen (smart-phones, tablets, etc) and data exploration.
3. Robust in terms of installation:
   The exhibition lasts seven months, the Mundaneum being open seven days a week.
4. Accessible for fast Content Management:
   Content could be updated by the Mundaneum staff without the need of nay technical person.

We will further investigate user reactions in museums but also in other applications like TV control while sitting or standing, alone or with other users, etc.

3D gestures recognition will also be integrated to send specific events to any interface providing linked data to the users.

Acknowledgments. Numediart is a long-term research program centered on Digital Media Arts, funded by Région Wallonne, Belgium (grant N°716631). This work has been partly supported by Communauté Wallonie-Bruxelles under the Research Action ARC-OLIMP (grant N°AUWB-2008-12-FPMs11).

References

1. OpenNI library, `http://www.openni.org`
2. Processing: A java-based programming tool, `http://www.processing.org`
3. Numediart Institute/ UMONS, ProcesSwing, `http://www.numediart.org/tools`
4. KinVi 3D: A Kinect-Enabled Virtual Interface Gadget for Windows Control, `http://www.kinvi3d.net/`
5. Eclipse, SWT: The Standard Widget Toolkit, `http://www.eclipse.org/swt/`