A 3D Interface to Explore and Manipulate multi-scale Virtual Scenes using the Leap Motion Controller

Bruno Fanini
CNR ITABC
Rome, Italy
bruno.fanini@gmail.com

Abstract — **Gesture-based interaction models can be efficient and simpler to understand if designed to correspond to common user interactions with the physical world. This paper presents a 3D interface and its implementation to quickly perform navigation and manipulation tasks in multi-scale and multi-resolution 3D scenes using a low-cost consumer sensor: the Leap Motion controller. The developed system has the goal of exploring the potential of accurate hands and fingers tracking alongside mid-air 3D gestures, to investigate specific design advantages and issues they present in such complex environments.**

Keywords — Gesture-based interaction; 3D manipulation; real-time; multi-scale

I. Introduction and Related Work

Interaction and navigation within complex 3D virtual contexts, in conjunction with a well-designed graphical user interface (GUI), are crucial to final user experience. Mouse-based GUIs have proven their robustness and flexibility, but a major shift towards "natural" user interfaces (NUIs) during the latest years is strong: this is not only related to research, but to applications and tools targeting broader consumer audiences as well.

A 3D interface is "*a User Interface that involves 3D interaction*" and 3D interaction can be defined as a layer that allows the user to perform different tasks *directly* in a 3D spatial context [9]. A widely used approach for 3D interfaces involves a physical 3D space used as input for the application: user provides such input by making gestures in this space. Typically, the software application is equipped with a gesture recognition system through the use of a sensor (e.g., Kinect [7] and others) allowing to map such physical movements into a set of predefined functionalities and actions. Unlike classic devices such as keyboard or mouse, these sensors allow to *spatially track* specific joints of a human body (e.g., arms, shoulders, hands, fingers, etc.) within a 3D context, providing for instance 3D positions <x,y,z>, motion data and orientations of different features. On the application side, the developer has to provide a software layer to process incoming data and recognize specific patterns (gestures) over an observed time slice, transforming these spatial inputs into direct actions inside the virtual 3D environment.

These devices and their progress in sensor accuracy, speed and efficiency [2][19] during the last years are laying solid foundations to deploy astonishing interaction models. Furthermore, low-cost systems and devices to detect hand [8] and body gestures [7] are nowadays becoming widely available to consumer market. Obviously, there are limitations of such tracking controllers that have to be considered for the design of efficient interaction models, such as device accuracy, noise issues and lighting conditions.

Some of the goals of 3D interfaces are to increase user engagement (for instance within serious gaming applications), application usability (natural mapping from 3D physical space to a 3D virtual space) and even reducing a few common bottlenecks related to 3D-oriented tasks [12]. Since 3D interaction is a quite recent topic, the maturity of 3D interface design principles lags behind those for standard GUIs. Given the wide range and diversity of input devices and interaction models, there isn't actually an established standard for 3D User Interfaces. While general Human Computer Interaction (HCI) principles such as Nielsen's heuristics [1] still apply, they are not sufficient for designing a usable 3D user interface.

This paper will describe a 3D interface design and its software implementation applied to a recent consumer device: the Leap Motion controller. The developed system is designed to explore and manipulate 3D objects in real-time within multi-scale and multi-resolution 3D virtual environments, using OpenSceneGraph framework as 3D visualization front-end. The following two sections will briefly introduce the device (II) and the OpenSceneGraph framework (III) while Section IV will describe the 3D interface and a test case where the system was applied.

II. The Leap Motion controller

The Leap Motion controller [8] is a small and inexpensive motion sensor available to consumer market since July 2013, composed of 2 small cameras and 3 infrared LEDs (Light-Emitting Diode) able to track hands, fingers and a few tools in mid-air inside a specific field of view, with sub-millimeter accuracy [2]. This 8 cm long USB (Universal Serial Bus) peripheral device (Figure 1) is designed to be placed on a physical desktop, facing upward. It operates in an intimate proximity, with a very high tracking frame-rate (when lighting conditions are optimal) inside a field of view of the shape of an inverted pyramid, centered on the controller.

 ISBN: 978-1-61208-325-4

Figure 1. The Leap Motion controller.

Several tests on the device report an effective range of 25 mm to 600 mm above the controller, where hands, fingers and tools are detected in mid-air with 3D positions, gestures, and other motion data. The growing community of developers around the Leap Motion has access to the latest Software Development Kit (SDK), Application Programming Interface (API) documentation and forums. The provided API allows to listen and report real-time data regarding hands (palm position, normal, direction, etc.) fingers (tip position, direction, etc.) and tools, alongside a few built-in recognized gestures such as circle, swipe, tap gestures, and a few others.

III. THE OPENSCENEGRAPH FRAMEWORK

OpenSceneGraph (OSG) [14] is an open source 3D *rendering middleware* and one of the world's leading scene graph API [3] used by application developers in fields of visual simulation, computer games, virtual reality, scientific visualization and modeling. The OpenSceneGraph framework is widely used within real-time 3D scientific visualization contexts due to its performance, portability and scalability, providing a huge set of functionalities that won't be discussed in detail in this paper. It is based on scene-graph structures, thus allowing the definition of spatial and logical relationships among different 3D models (nodes) in a virtual scene, specifically efficient on large and multi-resolution datasets. It allows to develop real-time 3D applications providing:

- Object-oriented functionalities
- Transformation nodes
- Loading of common 3D formats (Alias Wavefront OBJ [16], Autodesk 3D Studio Max [17], COLLADA [18], etc.)
- Management of large 3D environments, using spatial segmentation of the virtual world
- Remote node loading (via URL)
- Efficient management of level of detail (LoD)
- Instancing
- Paging

Transformation node in OSG is particularly useful to manage an object disposition, since it encapsulates a matrix transformation (position, rotation, scale) that is being applied to the entire sub-graph. Multi-resolution datasets are also fully supported and use appropriate representation (Level of Detail) depending on current camera view, while instancing techniques are able to reduce memory footprint through node sharing. Paging mechanisms in OpenSceneGraph allow scene portions (or "*pages*") to be loaded and unloaded at run-time from the main scene-graph, reducing system workload and GPU load, depending on current point of view and frustum.

Figure 2. Some examples of large-scale and multi-resolution 3D scenes in real-time using Front-Ends based on OpenSceneGraph framework.

Several test cases and projects (Figure 2) demonstrated the framework efficiency and performance specifically in handling and visualizing large, complex and multi-resolution datasets such as terrains, cities, etc. [4][5]. These are a few reasons that led to the framework choice as Front-End of 3D visualization in the current implementation of the proposed interface.

 ISBN: 978-1-61208-325-4

IV. 3D Interface Design and Implementation

This section will describe the 3D interface with its components, functionalities and overall design for navigation and manipulation in a multi-scale, multi-resolution 3D virtual environment using the Leap Motion controller. The proposed 3D interface is based on a two-handed input design [11][15]: in fact, one of the goals is to provide an efficient and fast interaction model for 3D-oriented tasks [12].

The whole concept takes advantage of the device accuracy, although several tests during design and development of first prototypes exhibited some data noise in special conditions, such as when hands approach the sides of the controller field-of-view. These issues were mostly solved at software level by applying special smoothing filters to received data, in this case to the features specifically involved in the design, such as palms and fingers. For completeness, a few definitions are provided:

1. *Virtual World (W)* encapsulates the scene-graph of the world (the whole scene or "global space"), potentially composed by complex/multi-scale 3D datasets.
2. *Manipulables* represent a collection of nodes (3D objects) subset of the virtual world W, having the property of being "editable" and able to be transformed over the time.
3. *View Configuration (v)* is composed by position (eye: <x,y,z>) and orientation (quaternion: <x,y,z,w>) representing a camera view or point of view into the current virtual world.
4. *Transition:* $v_t \rightarrow v_{t+1}$ where v is a View Configuration that changes through user interaction.
5. *Interaction Space (IS)* represents the 3D manipulation domain. It's located into the global space (W) through a transformation T_{is} (position, rotation and scale) and maps the physical space above the Leap controller.

It is important to mention that virtual world *W* and *manipulables* collection can rely both on local and remote locations (or mixed) allowing very interesting scenarios for the interface applied to any OpenSceneGraph-based application.

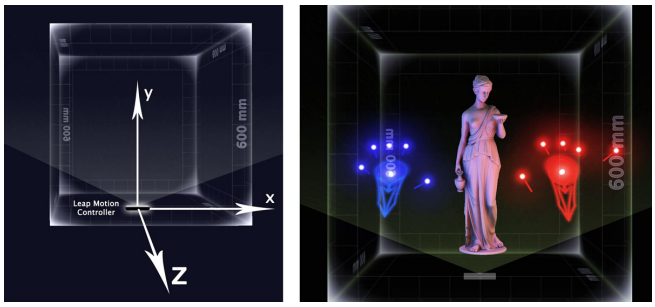

Figure 3. Interaction Space (IS) coordinate system and 3D visualization of both hands, fingers and device field-of-view (bottom).

The implemented system consists of a real-time visualization *Front-End*, a gesture listener and a 3D scene manager. Recognized features by the device, such as hands and fingers, are visualized and directly mapped inside the IS using a right-handed Cartesian coordinate system (Figure 3) with origin centered on the middle IR LED of the device. The rightmost hand in the developed system is represented in red while the leftmost hand in blue. Both hands are luminescent to provide customized real-time lighting of 3D models. The IS cube (600 x 600 x 600 mm) represents a reference of the physical space above the controller inside the virtual space.

The system provides the fundamental functionality to map local IS coordinates <x,y,z> (Leap Space) to World coordinates $<x_w,y_w,z_w>$ (World Space) and viceversa (1), for instance mapping a fingertip into world (global scene) coordinates.

$$<x_w,y_w,z_w> = T_{is}(<x,y,z>) \quad (1)$$

The next sub-section will describe the navigation model, how it influences T_{is} and when.

A. Navigation

A fulfilling exploration of a virtual world is a complex topic and requires special attention. Using standard peripherals, there are several well-established and familiar navigation models, for instance *pan-rotate-zoom* using the mouse, just to name one. When dealing with large virtual environments, there are even additional issues that need to be addressed. Content rarefaction in particular, is a typically disorienting aspect, although some solutions based on hotspots affordance have been proposed to reduce this phenomenon [13]. In such context, a 3D interface can provide a more efficient navigation model, allowing to combine different actions at the same time using a single gesture (e.g., pan action + zoom action, etc.) although sensor accuracy clearly plays a crucial role in this scenario. In general, each exploration session *E* can be represented as:

$$v = E(t). \quad (2)$$

where v is a varying *View Configuration* over time, depending on user input. Let us suppose the user starts from a large-scale context, wanting to focus on a detail of a small-scale 3D object to perform some task: how much time is required before the user reaches a satisfying view configuration v_f ? One of the goals of the proposed 3D interface is to minimize the t in (2) required for v_f.

The developed navigation model consists of two distinct phases: *stop* and *drag*. When the user is in stop phase, the system simply visualizes both hands and fingers inside the Interaction Space, listening for gestures. The user has a direct mapping of his own hands inside the manipulation domain since the IS is fixed in global space W (T_{is} is not changed) and aligned with the current View Configuration

(fixed). When the drag gesture is recognized, user is able to fully manipulate current View Configuration by "*dragging*" the space above the Leap Motion controller using both hands simultaneously, leading to a fluid bi-manual camera control model [6] in 3D. Previous work in literature [15] indeed suggests that specific tasks related to two-handed input can be performed effectively with a symmetric assignment of roles to both hands: in this case, camera motion and target are controlled by the position of both palms in 3D space. The system allows to combine into a single gesture a 5-*DoF* camera manipulation and *3D scale*. Denoting P_{left} and P_{right} as positions <x,y,z> of both palms in IS and C as the center between the two palms:

- *Left/Right, Up/Down* and *Forward/Back* are controlled by corresponding position of C relatively to IS.
- *Yaw* is controlled by palms difference along the IS z-axis.
- *Pitch* is controlled by average palms normal.
- *3D Scale* is controlled by palms distance and Target spot. This feature, similar to the *pinch* gesture on 2D multi-touch devices, provides the multi-scale exploration.

Scale manipulation feedback is provided by the radius of a ring overlay element that shows the relative scaling factor being applied and the target spot C', according to the center of the ring C. During the whole phase, IS is not changed: this design provides the user with a visual reference of the old IS state, useful in a multi-scale context since it enhances scale perception during camera transitions. For instance, increasing palms distance when a drag gesture is recognized, will shrink the IS and consequently scale the virtual hands in comparison to the world scene W.

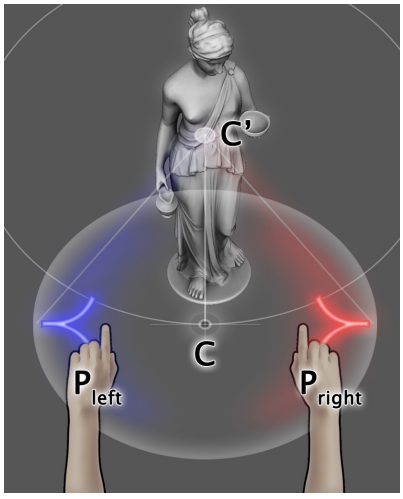

Figure 4. Drag gesture and target spot C' used in navigation model.

When dragging gesture ends (release), a 3D target spot C' is directly picked through C using an intersection ray on 3D geometry, as illustrated in Figure 4: the target spot is used to provide a new *View Configuration* and thus a new suitable IS. The system performs a validation of maximum and minimum length values of the segment $\overline{C\text{-}C'}$ to limit maximum and minimum scale, respectively. At this point, the user is once again in a stop phase and can iterate the whole process. Figure 5 illustrates a fast sequence of drag and stop to reach a partially occluded spot in a multi-resolution 3D model: starting from a stop phase (1) with no hands above the controller and a sample 3D model inside the IS, a *drag* gesture is initiated (2).

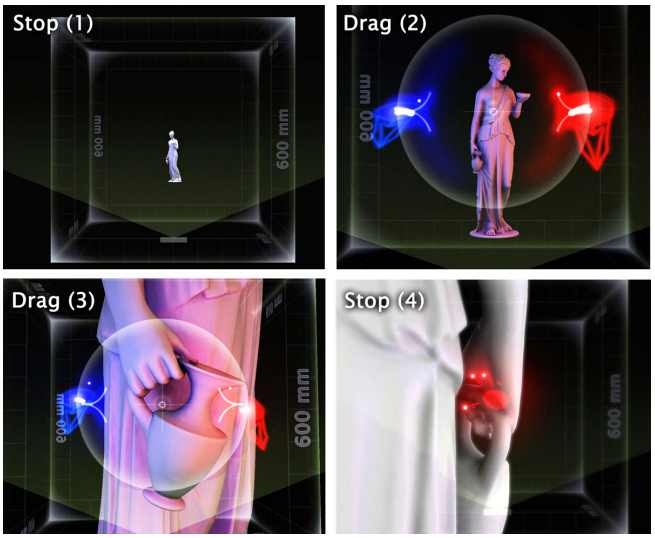

Figure 5. Stop and drag navigation model.

Notice how in (3) the center point of the ring is used to aim at the desired 3D target spot and in (4) the rightmost hand is partially intersecting domain geometries during a stop phase inside a new IS, with different scale. The cumulative nature of this approach aims to provide precise control by taking advantage of the Leap Motion device sub-millimeter accuracy [2], quickly alternating *stop* and *drag* phases to reach complex View Configurations in short times, maximizing user control during a transition $v_t \rightarrow v_{t+1}$. The scheme shown in Figure 6 is a graph of stop and drag sequences recorded during a test session on a sample 3D scene and their relation with Interaction Space scale magnitude over time. Stop sequences have variable durations since user is observing scene details, operating on IS domain (further details in the next Section) or resting (hands exit the device field-of-view). The phase alternation has also the goal of reducing fatigue for arms and hands [9][10]. The graph focuses on scale magnitude: notice how T_{is} is computed only at the beginning of a stop phase and at the end of a drag phase. In the recorded test-case, user session started from a scene overview (a) with IS mapped on a scale magnitude of 10 mt, moving in to focus on a detail (b) at a scale magnitude of 1 cm, then scaling up over one meter magnitude (c) and then focusing on a different 3D detail on 1 mm scale (d). The

 ISBN: 978-1-61208-325-4

system also allows to keep track of Interaction Spaces over the session: this is also useful to roll back to the previous IS with a simple gesture.

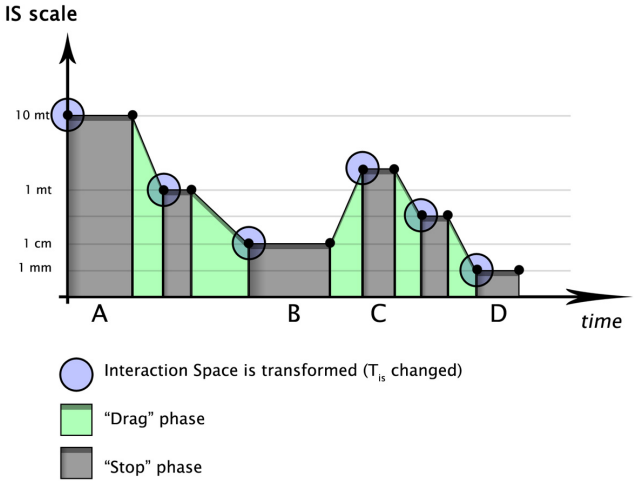

Figure 6. Recorded graph of Interaction Space magnitude (scale) over time.

The user is also able to record a specific View Configuration v_r for later recall, so that current pair $<v_r$, *scale*> can be stored on demand. The existence of *stop* phase in the proposed interface is further explained in the following sub-section.

B. Manipulation

When the user is not adjusting the view configuration through a *drag* phase, the system is in *stop* phase: hands are free to move inside the Interaction Space and what is currently contained. The design is specifically conceived to leave room for object manipulation tasks and other operations within current domain. During the stop phase, the user can activate an *edit* mode: this mode allows to apply several actions to items belonging to *manipulables* collection. Once the edit mode is activated by using a specific gesture and a visual feedback, user is able to operate on current domain and 3D objects inside the IS applying different transformations. Red hand (*dominant*) is used to grab a 3D object and move it using the direct IS mapping from physical space above the device and the virtual world. A selector is used when user is hovering a manipulable item to provide visual feedback. Object manipulation is thus performed by grabbing the virtual item and by moving it within the current Interaction Space (and its current scale). Grabbing gesture is recognized by the system on highlighted item when hand contraction is over a specified threshold and the item is picked up. When detected inside the IS, the blue hand (*non-dominant*) is used to apply additional transformations (rotation, scale, etc.) to the current selection. Like the proposed navigation model, in this case, bi-manual editing allows to apply multiple transformations at the same time (e.g., rotating and translating a picked object). The user can also take advantage of the multi-scale concept of the interface by getting closer to a spot and by operating on 3D objects using fine grained transformations. For instance, moving an entire house over a multi-resolution terrain and then zooming in to place a spoon on a table, since transformation accuracy is tied to IS scale.

C. Basic arrangement experiment

After the initial prototype, a test case was created using a basic scene consisting of a ground and a sample collection of *manipulables* (Figure 7): table, chair, armchair and a lamp. In order to test and verify basic usability of the system, a few internal lab sessions were carried out: users were asked to create plausible items arrangements through the developed system, starting from a randomized disposition (a). Initial observation sessions related to manipulation tasks provided sufficient data to modify the visual feedback through a white selector (b), that is shown only when the user hovers the item using the dominant hand (red). While the grabbed item is moved, the user can apply yaw rotation using non-dominant hand (c), finally reaching a plausible disposition of the scene (d). A second test case was performed to test navigation and manipulation of remote 3D datasets: the user was able to operate on grass, flowers and trees, streamed over the net (Figure 8).

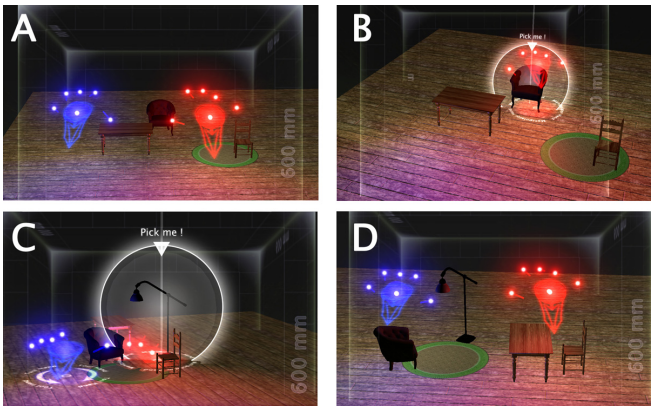

Figure 7. Test case for manipulation of 3D objects inside the Interaction Space.

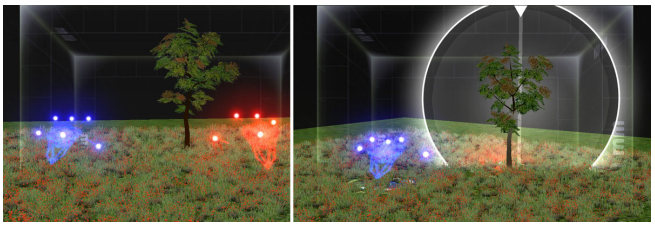

Figure 8. Manipulation and navigation of remote 3D datasets

These internal experiments and observations allowed to test and increase basic usability of the system by:

- Adding visual feedback on manipulable hovering (3D overlay selector).
- Adding constraints to navigation by limiting minimum-maximum IS scale inside a specific range and by enclosing allowable target spots (drag phase) inside global *manipulables* bounding-box.
- Adding explanatory welcome panels.

The same tasks were performed on the test scene using a mouse-based interface, providing participants with basic translation and rotation features.

V. Conclusion and Future Work

A 3D interface and its implementation were presented to explore and manipulate multi-scale and multi-resolution virtual environments. Although few experimental tests and observations were carried out on a limited number of users to evaluate interface usability, the following results can be highlighted:

- A navigation and manipulation model operating on multi-scale virtual environments.
- Efficient and interactive visualization on large, complex and even remote 3D datasets provided through the *OpenSceneGraph* framework integration [3][4].
- Use of a small and inexpensive desktop motion sensor (Leap Motion controller).
- A *Stop* and *Drag* design that maximizes the controller accuracy through relative *view configuration* transitions (Drag phase) and reduces arms fatigue during user sessions (Stop phase).
- Quick, multi-scale bi-manual manipulation (*Edit mode*) within the current IS through direct mapping.

The developed system that implements the 3D interface is based on the Leap Motion API and the OpenSceneGraph framework, allowing to be easily deployed to consumer market since the availability of the Leap Motion controller to the public. In order to obtain more accurate usability results, an adequate number of participants will be tested and more than one scene along with complex manipulation tasks will be provided. The integration with the OpenSceneGraph framework also provides portability across different Operating Systems (Windows, Linux, MacOS) and also inherits a large set of features provided by the framework, such as paging capabilities introduced in Section III for management of large scenes and manipulation of remote 3D scenes or sub-graphs (server-based approach).

The concept and design of Interaction Space allows great scalability and further extension. For instance, attached toolboxes to Interaction Space have been tested to provide the user with a set of selectable functionalities (duplicate object, remove object, etc.). Further advancements will be oriented to extend the manipulation functionalities and to introduce more constraints, such as axis snapping. The effectiveness of these additions will be supported by user testing, observations and experiments.

References

[1] J. Nielsen and R. Molich, “Heuristic evaluation of user interfaces”, Proceedings of CHI 90, pp. 249-256. New York, NY: ACM, 1990.

[2] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, "Analysis of the accuracy and robustness of the Leap Motion controller", Sensors 2013, 13, pp. 6380-6393, ISSN 1424-8220, 2013.

[3] W. Rui and Q. Xuelei, "OpenSceneGraph 3.0", Packt Publishing 2010 (eBook).

[4] B. Fanini, L. Calori, D. Ferdani, and S. Pescarin, "Interactive 3D landscapes online", International Archive of Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVIII-5/W16, 2011.

[5] X. Li, D. Kong, Y. Zhang, and Y. Sun, "Mass Models Dynamic Loading Method Research and Implementation based on OpenSceneGraph", Journal of Information & Computational Science 8: 2 (2011), pp. 362-369, 2011.

[6] R. Balakrishnan and G. Kurtenbach, "Exploring Bimanual Camera Control and Object Manipulation in 3D Graphics Interfaces", Proceedings of the ACM CHI 99 Human Factors in Computing Systems Conference, pp. 56-63, 1999.

[7] Microsoft Kinect (Windows): http://www.microsoft.com/en-us/kinectforwindows. Last accessed: 24th November 2013

[8] Leap Motion controller: https://www.leapmotion.com. Last accessed: 24th November 2013

[9] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev, "3D User Interfaces – Theory and Practice". Addison Wesley Longman Publishing Co., Inc., Redwood City, CA (USA), 2005.

[10] M. R. Mine, "Virtual environment interaction techniques", Tech. report, Chapel Hill, NC (USA), 1995

[11] W. Buxton and B. Myers, "A study in two-handed input", Proceedings of CHI 1986, New York, NY (USA) ACM, pp. 321-326, 1986.

[12] M. W. Gribnau and J. M. Hennessey, "Comparing single- and two-handed 3D input for a 3D object assembly task", Proceedings of CHI 1998, New York, NY (USA) ACM, pp. 233-234, 1998.

[13] B. Fanini, "Development and Evolution of Natural Interfaces in Virtual Heritage Applications", MIMOS Proceedings, 2012

[14] OpenSceneGraph: http://www.openscenegraph.org. Last accessed: 24th November 2013

[15] R. Balakrishnan and K. Hinckley, "Symmetric bimanual interaction", Proceedings of the SIGCHI conference on Human Factors in Computing Systems, ACM New York, NY (USA), ISBN:1-58113-216-6, pp. 33-40, 2000

[16] Alias/Wavefront OBJ file format: http://www.martinreddy.net/gfx/3d/OBJ.spec

[17] 3D Studio Max file format: http://www.fileformat.info/format/3ds/egff.htm

[18] COLLADA 3D model format: https://collada.org/

[19] K. Khoshelham, "Accuracy Analysis of Kinect Depth Data", International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XXXVIII-5/W12, 2011

 ISBN: 978-1-61208-325-4