Comparison of Multiple 3D Rotation Methods

Yao Jun Zhao, Dmitri Shuralyov, Wolfgang Stuerzlinger
Department of Computer Science and Engineering
York University, Toronto, Canada
yjzhao@cse.yorku.ca, shuryork@cse.yorku.ca, wolfgang@cse.yorku.ca

***Abstract*— In this paper, we present an experimental comparison of 3D rotation techniques. In all techniques, the third degree of freedom is controlled by the mouse-wheel. The investigated techniques are Bell's Trackball, Shoemake's Arcball and the Two-axis Valuator method. The result from a pilot showed no performance or accuracy difference among these three. However, we did observe minor differences in an experiment with more participants and trials, though these differences were not significant. Also, from questionnaires, we found that most of the users considered the use of mouse wheel helpful for completing the tasks.**

Keywords- 3D User Interface; 3D Rotation

I. INTRODUCTION

A. Motivation

Rotation of 3D objects with a 2D mouse is a typical task in computer-aided design and desktop virtual reality. However, the lack of intuitiveness in these 3D rotation techniques limit their use to a minority of users and only to selected tasks [1]. Previously presented techniques map mouse motions on (or near to) the object to 2D rotations of the manipulated object and motions away from the object to the third degree of rotation. By making the third degree of rotation directly accessible via the mouse wheel, we aim to improve the usability of 3D rotation techniques and hence expand the pool of potential applications and the user population.

B. Previous work

Chen introduced the "Virtual Sphere Controller" as the first *Virtual Trackball* to simulate the mechanics of a physical 3D trackball that can freely rotate about any arbitrary axis in 3-space [1]. He also introduced the "continuous XY with Additional Z controller" [3], which was later called Two-axis Valuator Trackball by Bade *et al.* [1]. Chen showed that "Virtual Sphere Controller" performed better than "continuous XY with Additional Z controller" in terms of completion time, while both achieved similar accuracy. In the same year, Bell presented a better version of the Virtual Trackball, which features a smooth transition across inner and outer areas of the trackball [2]. Shoemake subsequently improved Chen's work and presented the Arcball method, which provides kinetic agreement between mouse motion and object rotation [8].

There are other user interface techniques for 3D rotations, such as view-based techniques, controller-based techniques, multiple-degree-of-freedom techniques [5]. The latter two techniques give the best performance without sacrificing precision [6]. Yet, mouse-based interaction techniques still remain the most frequently used 3D rotation techniques. Henrisen *et al.* suggested this is due to the prevalence of mouse device and the industrial standardization of mouse-based techniques [5], also via rotation widgets that control one degree of rotation at a time. Among the mouse-based interaction techniques that afford at control of at least two degrees of rotation simultaneously, Bade *et al.* showed that the Two-axis Valuator outperforms Bell's trackball and Shoemake's Arcball in terms of completion speed and user satisfaction in a "search and shoot" tasks [1]. However, one has to note that this task required only control of two degrees of rotation, i.e. the study did not involve a full 3D rotation task.

Gallo *et al.* introduced a user interface that featured a depth-enhanced mouse pointer and a novel rotation technique that uses the object geometry as the rotation handle. They showed that the utilization of geometry-based handle significantly improved the completion time over the Two-axis Valuator technique [4]. Interestingly they demonstrated this difference only for tasks requiring three degrees of rotation, but not for those demanding only two degrees of rotation.

Shuralyov and Stuerzlinger implemented a 3D desktop manipulation system called Slide [9], which uses the sliding algorithm first presented in SESAME [7]. In this system, any manipulated object is always in contact with other surfaces in the scene. This *sliding* or *snapping to surfaces* technique effectively reduces one degree of freedom from translations and thus makes it easy to control object positioning with the mouse. For 3D object rotations, they used the Two-axis Valuator technique and again ensured that the object was always in contact (but not in collision) with the surface it rests on. To support precise rotation alignment, the default setting is to snap the object to the nearest 30 degree multiple in all three Euler angles (*snapping to nearest angles*) in the global coordinate system, upon release of the mouse button [9]. We made the third degree of rotation for all three rotation techniques directly available via the mouse wheel, which this makes this operation more easily accessible to users and reduces mouse movement distances. Based on some initial experimentation, we set the increment for each mouse wheel event to 10 degrees for controlling the third degree of freedom for rotations.

C. Hypothesis

We expect that the use of the mouse wheel to control the third degree of freedom for rotations would improve performance and usability for all techniques, as this should reduce mouse movement times. This improvement may reduce the differences among the considered mouse-based 3D rotation techniques; or, it may significantly improve only specific technique(s), which will thus differentiate them more in terms of rotation performance, accuracy or user satisfactory.

Another consideration is the effect of "snapping to surfaces" on the rotation techniques. Snapping causes a rotated object to bounce up and down during the rotation tasks. This may yield negative effects to the rotation performance, accuracy and user satisfactory. However, we expected these negative effects would not cause big trouble since our tasks were relatively simple.

An additional issue is the potential effect of task difficulty. Likely, simpler tasks require less completion time and may result in better accuracy as stated in Chen's paper [3]. Finally, learning effects may also cause problems in our experiments, and we need to make sure to reduce them during the user study and the following analysis.

II. TECHNICAL APPROACH

In this paper, we compare three mouse-based rotation techniques, Bell's trackball, Shoemake's Arcball and Two-axis Valuator, under conditions with or without "Snapping to Surface".

A. Virtual Trackballs

Shoemake's Arcball [8] is a special case of Chen's trackball [3] in which p (a point on image plane) is the orthographic projection of P (a point in the full 3D scene) as shown in Figure 1.

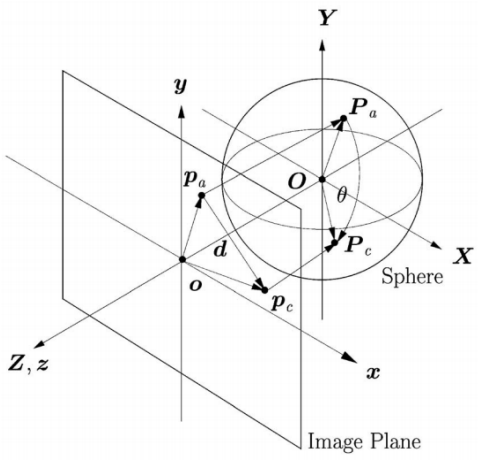

Figure 1. A virtual trackball can be thought of as a 3D sphere located behind the screen. The points $\vec{p}$ on the image plane are mapped to points $\vec{P}$ on the 3D sphere. Image reproduced from [5]

The function $m_{Shoemake}$ maps p to P by

$$m_{Shoemake}(p) = m_{Shoemake}(x, y, 0) = P$$

$$= \begin{cases} \begin{pmatrix} x \\ y \\ \sqrt{r^2 - (x^2 + y^2)} \end{pmatrix} & if\ \sqrt{x^2 + y^2} \le r \\ \frac{r}{\sqrt{x^2+y^2}} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} & if\ \sqrt{x^2 + y^2} > r \end{cases} \quad (1)$$

such that a point outside the projected sphere is mapped to the nearest point on the rim of the sphere [5]. Due to this, Shoemake's Arcball suffers from noticeable discontinuities in some situations when the mouse moves into the projection of the sphere from the outside [5].

Instead of mapping points onto a sphere as in Shoemake's Arcball, Bell's trackball maps them onto a surface combining a sphere and a hyperbola, such that, if a 2D point is close to the center of the trackball, the surface is a sphere, otherwise, it is a hyperbola [5]. The function m_{Bell} is,

$$m_{Bell}(p) = m_{Bell}(x, y, 0) = P$$

$$= \begin{cases} \begin{pmatrix} x \\ y \\ \sqrt{r^2 - (x^2 + y^2)} \end{pmatrix} & if\ \sqrt{x^2 + y^2} \le \frac{r}{\sqrt{2}} \\ \begin{pmatrix} x \\ y \\ \frac{r}{2\sqrt{x^2+y^2}} \end{pmatrix} & if\ \sqrt{x^2 + y^2} > \frac{r}{\sqrt{2}} \end{cases} \quad (2)$$

Compared to Shoemake's Arcball, Bell's trackball offers more smooth rotations since the orientation of the axis of rotation is a continuous function of the screen coordinates. However, this smoothing treatment in Bell's trackball sacrifices the predictability of a simulated physical ball since the ball is no longer rotating along the great circle arc defined by the Origin O, P_a and P_c when $\sqrt{x^2 + y^2} > \frac{r}{\sqrt{2}}$.

The Two-axis Valuator trackball, maps mouse motion to two axes of the view coordinate system, both orthogonal to the view-vector of the camera through the center of the object [9]. Namely, horizontal mouse movement is mapped to rotations about the up-vector of the camera and vertical mouse movement is mapped to a rotation about the vector perpendicular to the up-vector and the view-vector. Then diagonal mouse movements are mapped to a combined rotation about both axes. This makes the Two-axis Valuator very predictable. However, there is no explicit provision for rotation about the view-vector. In the "continuous XY with Additional Z controller", Chen *et al.* define mouse movement in the areas outside the sphere as view-vector rotation and use the Two-axis rotations only inside the trackball [3]. The approach we implemented uses the mouse wheel to achieve view-vector rotation. Thus two-axis rotations are no longer bounded. The benefit of this is that rotations can be performed without concern for the starting position of the cursor, which is a requirement for either Bell's or Shoemake's trackballs. Also, the user can rotate the object about an axis for *a full revolution or more* by a single drag.

B. Snapping to surfaces

The "Snapping to surfaces" or SESAME sliding algorithm implements the idea of always maintaining contact between the

manipulated object and (static) other surfaces in the scene. While object is translated, it slides on these surfaces stably under the mouse cursor. With this, only 2D input is necessary to specify a 3D position, since the object can only move on the 2D manifold of the visible surfaces. This feature is achieved in Slide through continuous collision detection with Opcode 1.3, which also prevents object interpenetration.

The sliding algorithm uses the frame buffer to detect all surfaces behind the current selected object. During translation, the object is snapped in the view direction to the nearest surface behind it and slides on the corresponding background face, edge, or vertex. Using the first surface behind the object ensures that this algorithm does not always "pop to the front". Hence, user can slide an object under another. However, if the object disappears completely behind another object, i.e., no part of moving object is visible, the system "pops" the moving object to the front, which ensures the least suprise. Meanwhile, perspective correction ensures that the object remains stable under the cursor.

Similarly, while an object is rotated, it remains in contact with the current surface it is resting on, via the above-mentioned frame-buffer method with the view direction set to the contact normal. This results in the center of the object moving up and down relative to the contact plane [9], but avoids object interpenetration, which we find helpful for novices.

C. *The mouse wheel to control the third degree of freedom*

As stated in section II.*A*, it is necessary to provide some control method for the third degree of freedom in rotations. This is inherent to the Two-axis Valuator method, which is one of the reasons why we assign the mouse wheel to this task. For Bell's trackball and Shoemake's Arcball this is not the case as these two methods map circling the mouse outside of the trackball widget to rotation around the view-vector. This permits this method to rotate objects freely to any orientation in 3D space without any additional control method. However, we believe that using the mouse wheel to control that third degree of rotation makes both trackballs easier to use and can improve user satisfaction. With this, users can perform rotation around the view-vector without regard for the location of the cursor, i.e., users do not need to move the cursor out of the trackball to achieve this rotation.

Since mouse wheel input is discrete, one needs to find an appropriate incremental change angle for each mouse wheel "click" forward and backwards. On the one hand too small an increment might induce excessive workload and thus reduce usability. On the other hand, too large an increment might jeopardize the precision user can achieve. For our user study, we used a moderate angle increment of 10 degrees for each mouse wheel event.

D. *Implementations*

The system is based on Shuralyov and Stuerzlinger's Slide system [9]. This system provides many features beneficial to 3D manipulation, such as keeping objects always in contact and collision detection. More precisely, we used objects from a 3D puzzle assembly task in this study. The "snapping to surfaces" functionality was enabled only for certain conditions. The original implemented rotation technique in this system was the Two-axis Valuator method where the third degree of freedom is controlled with the mouse wheel. For the comparison of the three rotation techniques, we added implementations of Bell's trackball and Shoemake's Arcball based on Henriksen [5]. For both these trackball methods the third degree of rotation freedom was also controlled with the mouse wheel.

A desktop PC with 2.93GHz Intel i7 CPU, GeForce GTX 460 graphics card and 4 GB memory was used for user study. The input device was a Microsoft Basic Optical Mouse with two buttons and a middle wheel. We used a 23" 1920x1080 LCD monitor for output.

III. EMPIRICAL EVALUATION

A. *Experimental design (user study)*

Our experiment was designed to compare the participants' performance using Bell's trackball, Shoemake's Arcball and the Two-axis Valuator method. The main performance measures recorded were completion time and accuracy. The participants were given short instruction in the use of each technique during the 10-minute training sessions. To help participants to develop a good mental model for each technique we also displayed appropriate visual feedback, namely either a "ball" or axes, as appropriate for the particular method. For Bell's trackball and Shoemake's Arcball we showed three orthogonal circles around the objects (painted red for Bell's and blue for Shoemake's method). For the Two-axis Valuate method we showed two perpendicular axes through the center of the object, as shown in Figure 2.

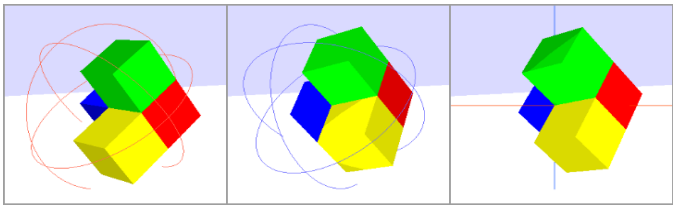

Figure 2. Visual cues for different rotation techniques, from left to right: (1) Bell's trackball, (2) Shoemake's Arcball and (3) Two-axis Valuator

The task was similar to the rotation matching tasks in Chen *et al* [3] and Hinckley *et al* [6]. In each task, a participant clicked the left mouse button to start a trial. Then, the participant was given an initial orientation of a puzzle block and an image of the target orientation in a window on the left side, as shown in Figure 3. The participant dragged the cursor to rotate the block while pressing the right mouse button. At any given time, the participant could use the middle wheel to rotate the block clockwise or counterclockwise about the view axis regardless whether they were pressing the right mouse button or not. This enabled access to all three degrees of rotation simultaneously. After the participant visually confirmed they obtained the best possible match, he/she pressed the space bar to terminate the current trial.

Tasks were performed under two independent factors: rotation techniques (three) and snapping settings (with and without "snapping to surfaces"). Hence, a total of six

conditions were investigated. In each condition, four different tasks were repeated five times. The order of conditions was determined via a Latin Square to eliminate learning effects.

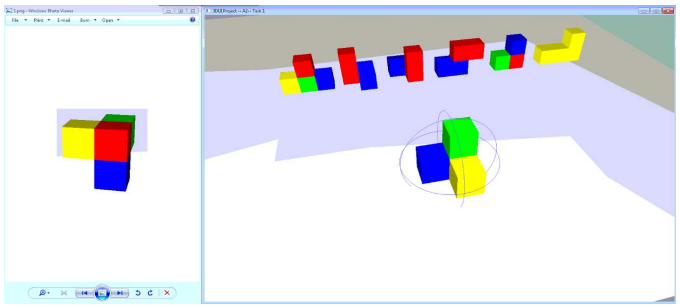

Figure 3. The user study environment: the target orientation image is placed to the left of the 3D scene.

The four tasks involved four different puzzle blocks: the Sputnik-shape, T-shape, Snake-shape, and mirrored-Snake-shape. The rotations required to bring them into the right position are shown in Table I. The mean required quaternion angle was 164.46 degrees. The mean required Euler's angles were 94.39, 41.95 and 92.61 for yaw, pitch and roll, respectively.

We recruited twelve participants. All were right-handed, with nine male and three female with age ranging from 15 to 40 (8 of them were in their twenties). Only two of them played computer or video games for more than 5 hours per week. Most of them were from the local student population. They were asked to finish the tasks as fast as possible and match the target orientations as closely as possible. The participants were also informed that the expected time to finish each task was about 20 seconds and they could take a short break between tasks.

TABLE I. REQUIRED ROTATION ANGLES FOR THE FOUR TASKS

	Sputnik	**T-shape**	**Snake**	**Mirrored-Snake**
Quaternion	147.12	197.85	179.42	169.15
Yaw	85.59	20.02	175.98	95.98
Pitch	18.07	53.64	26.70	69.39
Roll	117.79	149.54	14.33	88.77

Angle values are all absolute values in degrees

After the participants finished all the tasks, they were asked to fill in questionnaires about the Ease of Use, Performance, Predictability, Accuracy and Overall Usability among the three techniques. We also polled their opinion about the usefulness of using the mouse wheel to provide access to the third degree of rotation freedom. In the end, the participants were asked for general comments. We also collected basic personal information, such as includes age, gender, and time spent on computer usage as well as computer games.

B. *Results*

The first trials took much longer than all other ones. Hence, we removed these during the data processing step and report all results only for the last four trials.

The mean and standard deviation of completion time and errors under different conditions are shown in Figure 4. While there are no strong differences between the techniques, one can see that "snapping to surfaces" had a noticeable effect on completion time (Figure 4 A) but not on accuracy (Figure 4 B and C). To get better insights, we performed a statistical analysis of the data.

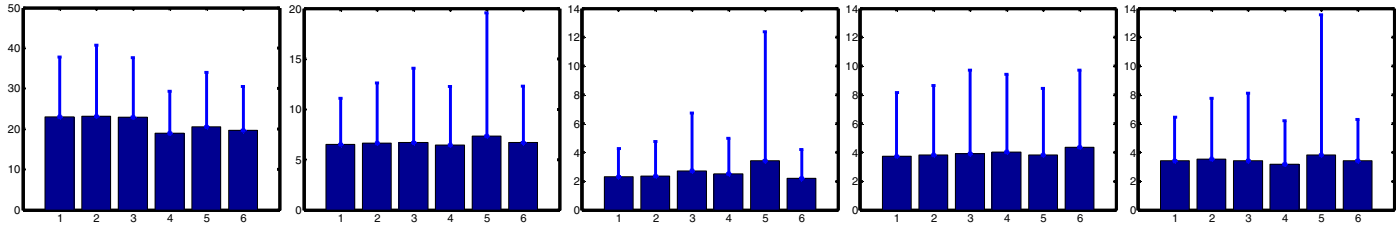

Figure 4. The results on completion time (A), error in quaternion angle (B), error in Euler's angles (C1-C3): (1)-(3) with Snapping to surface, (4)-(6) without Snapping to surface; the order of techniques in (1)-(3) or (4)-(5) is, Bell, Shoemake and Two-axis

IV. ANALYSIS AND DISCUSSION

ANOVA's were conducted on completion time, errors in quaternion angle and errors in Euler's angles. As one might expect, participant was a major factor that affected the results in all aspects. This is obvious since each individual had different rotation skills, mouse manipulation skills, and placed different priorities on speed vs. accuracy.

"Snapping to surfaces" had a significant effect on Completion Time $F_{1,11} = 34.24, p < 0.0001$, but not on Accuracy, likely, on quaternion angle error, $F_{1,11} = 0.43$, n. s., on yaw angle error $F_{1,11} = 0.98$, n. s., on pitch angle error, $F_{1,11} = 1.22$, n. s., and on roll angle error, $F_{1,11} = 0.01$, n. s. It took longer to complete tasks in the "snapping to surfaces" conditions, about 23.1 seconds on average compared to 19.7 seconds without snapping. This may due to the fact that participants spent extra time to adjust the rotation whenever the block jumped up or down due to the snapping functionality. Nevertheless, the accuracy was not affected by the vertical translations.

There was no significant learning effect on completion time $F_{3,33} = 1.55$, n. s., on quaternion angle error $F_{3,33} = 0.59$, n. s., on yaw angle error $F_{3,33} = 0.8$, n. s., on pitch angle error, $F_{3,33} = 0.48$, n. s., and on roll angle error, $F_{3,33} = 1.35$, n. s.

But Tasks had a significant effect on accuracy in terms of quaternion angle error, $F_{3,33} = 12.56, p < 0.0001$, and pitch angle error, $F_{3,33} = 43.07, p < 0.0001$.

We could not find statistical differences between the rotation techniques either in completion time or accuracy. But compared to a small pilot experiment, there is the potential for a trend that we may find significant difference among techniques as more trials are performed. But, it is also possible that adding the mouse wheel to control the third degree of freedom combined with the fact that rotations in all three degrees of freedom is harder, largely hides all potential differences between the techniques. In Bade's experiments [1], participants had to move the cursor outside of Bell's trackball or Showmake's Arcball widget to rotate an object clockwise or counterclockwise about the view vector. However, the same functionality with the Two-axis Valuator was accessed with a different form of control (mouse button or key). It is possible that this choice alone made the Two-axis Valuator more superior in terms of view-vector rotation. Note also that Bade's experiments involved only two degrees of freedom, as the orientation of the final result apparently did not matter. In Chen's experiments [3], participants were required to move the cursor out of each investigated trackball widget to rotate the objects around the view-vector. Consequently, the amount of rotation of the "continuous XY with Additional Z controller" was bounded in two dimensions by the extent of the spherical trackball widget. We suspect that this difference between Bade's and Chen's experiment is the most likely explanation for the different conclusions reached by the experiments (Bade concluded Two-axis Valuator as better method but Chen concluded the Virtual Sphere Controller as the best). In our experiment, all investigated techniques used the mouse wheel to control the third degree of freedom. But our Two-axis Valuator technique permitted unbounded. The improvement of all these techniques combined with the fact that full 3D rotations are known to be cognitively challenging might have erased all potential performance gaps among them, hence, resulted in a statistically insignificant difference in terms of completion time.

Another finding from ANOVA is that, by comparing the errors of three components of Euler's angles, we noticed that pitch errors were significantly larger than yaw and roll errors, $F_{2,33} = 29.87, p < 0.000$. This may be due to the uncommon use of up/down rotations of human beings in real life or as this rotation is less easily perceptible.

Based on the results of the questionnaires, 5 participants considered Bell's trackball, 4 considered Two-axis Valuator and only 2 considered Shoemake's Arcball as the easiest method. 6 participants felt Bell's trackball was the fastest approach while 3 decided for the Two-axis Valuator and 2 for Arcball. Also, 6 participants considered Bell's trackball best on predictability or controllability while there were 3 proponents of either technique in the rest. In terms of accuracy, 5 participants voted Two-axis Valuator, 3 voted Arcball and 3 voted Two-axis Valuator. For overall usability, 6 participants considered Bell's trackball as their preference while 3 preferred Two-axis Valuator and only 1 preferred Arcball. 11 of 12 participants regarded the third degree of freedom added by mouse wheel as Useful, and 5 "highly appreciated" it. However, 4 participants complained about the mouse wheel increment as being too large.

Using the mouse wheel to manipulate the third degree of freedom yielded high user satisfactions. However, the 10 degree increment for each mouse wheel step was not widely appreciated. One potential improvement is to use accelerated increments for mouse wheel rolling. That is, when a user rolls the mouse wheel slowly, the orientation increment is small, and when the user rolls the wheel quickly, the increment becomes larger, proportional to the rolling speed.

V. CONCLUSION

In our rotation matching experiment we did not find significant performance difference for the three investigated techniques. This may due to the use of the mouse wheel to control rotations around the view vector for all three investigated techniques and the removal of the interaction boundaries in the Two-axis Valuator method. We also found "snapping to surfaces" significantly decreased the efficiency of 3D rotations. Task difference was another significant factor in terms of rotation performance and accuracy.

Although most participants gave positive feedback on using the mouse wheel to control the third degree of freedom, this approach could be improved substantially. In the future, we plan to apply acceleration to the mouse wheel. Also, we plan to improve the tasks in a further study.

REFERENCES

[1] R. Bade, F. Ritter and B. Preim, Usability Comparison of Mouse-Based Interaction Techniques for Predictable 3d Rotation, Smart Graphics 2005, 138-150

[2] G. Bell, Bell's Trackball, http://members.tripod.com/professor_tom /index.html, 1988

[3] M. Chen, S.J. Mountford and A. Sellen, A Study in Interactive 3-D Rotation Using 2-D Control Devices, Computer Graphics, vol. 22, no. 4, pp. 121-129, Aug. 1988.

[4] L. Gallo, A. Minutolo, and G. De Pietro, A User Interface for VR-ready 3D Medical Imaging by Off-the-shelf Input Devices, Computer in Biology and Medicine, vol. 40, no. 3, pp. 350-358, 2010.

[5] K. Henrisen, J. Sporring and K. Hornbaek, Virtual Trackballs Revisited, IEEE Transaction on Visualization and Computer Graphics, 2004, 10(2), 206-216.

[6] K. Hinckley, J. Tullio, R. Pausch, D. Proffitt, N. Kassell, Usability Analysis of 3D Rotation Techniques, Proc. Of ACM UIST'97 (1997) 1-10.

[7] J.-Y. Oh, W. Stuerzlinger, J. Danahy, SESAME: Towards Better 3D Conceptual Design Systems, ACM DIS 2006, 80-89.

[8] K. Shoemake, Arcball: A User Interface for Specifying Three-Dimensional Orientation Using a Mouse, Proc. Graphics Interface'92, pp. 151-156, 1992.

[9] D. Shuralyov and W. Stuerzlinger, A 3D Desktop Puzzle Assembly System, IEEE 3D UI Symposium 2011, 141-142, March 2011.