Chapter 33
Interaction with 3D Environments Using Multi-Touch Screens

Francisco Ortego, Naphtali Rishe, Armando Barreto and Melek Adjouadi

Abstract The increase in availability of multi-touch devices has motivated us to consider interaction approaches outside the limitations associated with the use of a mouse. The problem that we try to solve is how to interact in a 3D world using a 2D surface multi-touch display. Before showing our proposed solution, we briefly review previous work in related fields that provided a framework for the development of our approach. Finally, we propose a set of multi-touch gestures and outline an experiment design for the evaluation of these forms of interaction.

33.1 Introduction

This paper presents the initial development of our approach to work with 3D data environments using a multi-touch display. We introduce our emerging methods in the context of important previous work, resulting in our proposed gesture recognition approach and definition of translation and rotation gestures for multi-touch

F. Ortego (✉) · N. Rishe
School of Computing and Information Sciences, Florida International University, Miami, FL, USA
e-mail: Forte007@fiu.edu

N. Rishe
e-mail: NDR@acm.org

A. Barreto · M. Adjouadi
Electrical and Computer Engineering Department, Florida International University, Miami, FL, USA
e-mail: BarretoA@fiu.edu

M. Adjouadi
e-mail: Adjouadi@fiu.edu

interaction with 3D worlds. In this same process, we try to answer two important questions and provide an evaluation path to be implemented in the near future.

3D navigation and manipulation are not new problems in Human–Computer Interaction (e.g., [1, 2]) as will be shown in our brief review of previous work. However, with the availability of multi-touch devices such as the iPad, iPhone and desktop multi-touch monitors (e.g., 3M M2256PW 22″ Multi-Touch Monitor) new concepts have developed in order to help the transition to a post-Windows-Icon-Menu-Pointer (WIMP) era. This gives rise to important questions such as: (1) What is the most appropriate mapping between the 2D interface surface and the 3D world? and (2) Can previous techniques used with other devices (e.g., joystick, keyboard and mouse) be used in 3D navigation?

To begin answering these questions, we first endeavor to understand touch interactions and previous multi-touch work. We believe that all those aspects create a foundation that is necessary for the development of a sound post-WIMP framework [3]. After the related work section, we cover our proposed solution, discussion and future work.

33.2 Background

33.2.1 Understanding Touch Interactions

A common option for multi-touch interaction is to use the set of points corresponding to *n* touches in a direct manner. However, to achieve a more natural interaction between the screen and the user, studies like [4–8] provide a different take on how touch information can be used. For example, [4] studies finger orientation for oblique touches which gives additional information without having extra sensors (e.g., left/right hand detection). In another example, Benko and Wilson [8] study dual finger interactions (e.g, dual finger selection, dual finger slider, etc.). Additional work dealing with contact shape and physics can be found in [6, 7]. In a very comprehensive review of finger input properties for multi-touch displays [5] provides suggestions that have been used already in [4].

One aspect that is important to have in mind is whether to keep rotations, translations and scaling separate [9] or combined [10]. If the latter is chosen, the user's ability to perform the operations separately may become a problem [9].

One very important point found in [11, 12] is that one-hand techniques are better for integral tasks (e.g., rotation) and two hands perform better with separable tasks. For our particular work, one can think of using one hand to perform common rotations and translations, and using two hands when special rotations need to be performed, utilizing the second hand to indicate a different point of reference.

33.2.2 Virtual Devices

In [1], Nielsen and Olsen used a triad mouse to emulate a 3D mouse. What is important about this work is how they perform 3D rotations, translations and scaling (one at a time). For example, in 3D rotation, they use point P_1 as the axis reference and points P_2 and P_3 to define the line forming the rotation angle to be applied to the object. In more recent work [10], one can find subtle similarities with [1], in the proposition of defining a point of reference to allow seamless rotation and translation.

The Virtual Sphere [2] is an important development in 3D rotation methods previously proposed, which was tested against other virtual control devices. It was found that the Virtual Sphere and the continuous XY + Z device behaved best for complex rotations (both devices behave similar with the exception that the XY + Z device does not allow for continuous rotations about all x, y, z axes). The Virtual Sphere simulates a real 3D trackball with the user moving left-right (x-axis), top-down (y-axis) and in circular fashion (z-axis) to control the rotation of the device. Similar work can be found in [13] with The Rolling Ball and in [14] with the Virtual Trackball. Another idea, similar to The Virtual Sphere [2] is the ARCBALL [15]. The ARCBALL "is based on the observation that there is a close connection between 3D rotations and spherical geometry" [16].

33.2.3 Gesture Recognition

Different methods for gesture recognition have been used in the past including Hidden Markov Models [17], finite state machines [18, 19], neural networks [20], featured-based classifiers [21], dynamic programming [22], template matching [23], *ad hoc* recognizer [24] and simple geometric recognizers [25–27]. An in-depth review can be found in [28]. For the purpose of this paper, we have concentrated in the simple geometric classifier, also known as geometric template matching.

The \$1 algorithm [25] provides a simple way to develop a basic gesture recognizer directed at people that do not have either the time or knowledge to implement more complicated algorithms. For example, algorithms based on Hidden Markov Models or neural networks. At the same time, \$1 provides a very fast solution to interactive gesture recognition with less than 100 lines of code [25]. The algorithm goes through four steps. The first step is to resample the points in the path. The idea is to make gestures comparable, by resampling each gesture at N points (e.g., $N = 64$) [25]. The second step is to find "the angle formed between the centroid of the gesture and gesture's first point" [25], which is called the indicative angle. Then this step of the algorithm rotates the gesture so the indicative angle is equal to zero. The third step includes the scaling of the gesture to a reference square which will help to rotate the gesture to its centroid. After scaling, the gesture is translated to a reference point so the centroid is at (0, 0). Finally, step 4 calculates the distance between the candidate gesture to each stored template and using a score formula which gives 0 to the closest gesture and 1 to farthest gesture [25]. The primary limitation of this algorithms are found in the incorrect processing of 1D gestures.

The $N algorithm [26] extends the $1 algorithm [25] primarily to allow single strokes to be recognized. The algorithm works by storing each multistroke as a unique permutation. This means that for each multistroke composed of two 2 strokes, the system creates 8 unistrokes. Another particular feature of this algorithm is that it allows the option for the stroke to be bounded by an arbitrary amount of rotation invariance. For example, to make a distinction between A and ∀, rotation must be bounded by less than ±90° [26]. This algorithm also supports automatic recognition between 1D and 2D gestures by using "the ratio of the sides of a gestured's oriented bounding box (MIN-SIDE vs. MAX-SIDE)" [26]. In addition, to better optimize the code, $N only recognizes a sub-set of the templates to process. This is done by determining if the start directions are similar, by computing the angle formed from the start point through the eight point. In general, this algorithm which contains 240 lines, was faster than $1 when using 20 to 30 templates. Algorithms $1 and $N utilized the Golden Section Search [29]

Other methods similar to $1 [25] and $N [26] algorithms have been implemented. For example, the *Protractor Gesture Recognizer* algorithm [27] works by applying a nearest neighbor approach. This algorithms is very close to the $1 algorithm [25] but attempts to remove different drawing speeds, different gesture locations on the screen and noise in gesture orientation.

33.2.4 Multi-Touch Techniques

We believe that all of the related work dealing with multi-touch, regardless whether it was designed for manipulation of objects or navigation of 3D graphical scenes, can contribute to the set of unifying ideas that serves as the basis for our approach.

Some of the work in 3D interactions has been specific for multi-touch, which is our focus as well. In [10], Hancock et al. provide algorithms for one, two and three-touches. This allows the user to have direct simultaneous rotation and translation. The values that are obtained from initial touches T_1, T_2 and T_3 and final touches T'_1, T'_2 and T'_3 are Δyaw, Δroll and Δpitch which are enough to perform the rotation in all three axes and Δx, Δy, Δz to perform the translation. A key part of their study showed that users prefer gestures that involve more simultaneous touches (except for translations). Using gestures involving three touches was always better for planar and spatial rotations [10].

A different approach is presented in Rotate 'N Translate (RNT) [30], which allows planar objects to be rotated and translated using opposing currents.

This particular algorithm is useful for planar objects and it has been used by 3D interaction methods (e.g., [31]).

A problem that occurs when dealing with any type of 3D interaction in multi-touch displays, is the one of spatial separability [9]. To address this problem, in [9], the authors proposed different techniques that will allow the user to perform the correct combination of transformations (e.g., scaling, translation + rotation, scaling + rotation + translation, etc.). The two methods that were most successful were Magnitude Filtering and First Touch Gesture Matching. Magnitude Filtering works similarly to snap-and-go [9]. This method has some differences from normal snapping techniques because it does not snap to pre-selected values or objects. In addition, the authors introduce a catch-up zone allowing "continuous transition between the snap zone and the unconstrained zone." [9]. The latter method, First Touch Gesture Matching, works by minimizing "the mean root square difference between the actual motion and a motion generated with a manipulation subset of each model" [9]. To select the most appropriate model, each prospective model creates two outputs, best-fit error and magnitude of the appropriate transformation. These outputs are given to an algorithm that decides which models to apply.

33.2.5 Design Guidelines

In [3], Jacob et al. proposed interfaces within a post-WIMP framework. In this framework, they try to find a balance between Reality Based Interactions (RBI) and artificial features. RBI includes Naïve Physics, Body Awareness and Skills, Environment Awareness and Skills and Social Awareness and Skills. To allow an application to balance itself, they proposed certain unrealistic features, providing a tradeoff between RBI and artificial methods. The characteristics that can be traded are: expressive power, efficiency, versatility, ergonomics, accessibility and practicality [3].

In another work [10], Hancock et al. also proposed some more specific guidelines when dealing with multi-touch rotations and translation. This includes the ability to provide more degrees of freedom than classical WIMP (ability to do rotation and scale independently or together), provide a constant connection between the visual feedback and the interaction, prevent cognitive disconnect by avoiding actions that the user may not be expecting, and provide realistic 3D visual feedback.

For a concise set of guides for 3D interaction, please review Bowman et al. [16].

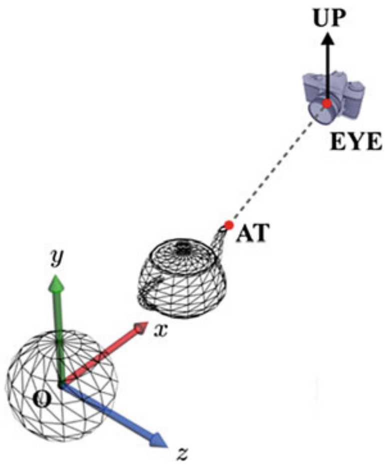

Fig. 33.1 Camera view [33]

33.3 Proposed Solutions

33.3.1 Set Up

33.3.1.1 Camera

To test our work, we use OpenGL and perform the visualization through a virtual camera developed in [32] and described by [33], as shown in Fig. 33.1. One can see that the UP vector indicates which way is up, the EYE (or ORIGIN) vector indicates the position of the camera and the AT (or FORWARD) vector indicates the direction in which the camera is pointing.

33.3.1.2 Multi-Touch

We are using Windows 7 multi-touch technology [34] to test our proposed solutions with a 3M M2256PW Multi-touch Display. Windows 7 provides two ways of using the touches from the device. The first one is gesture-based, identifying simple pre-defined gestures and the second is the raw touch mode which provides the ability to develop any type of interaction. We chose the latter because our end goal is to create custom user interactions and for that we preferred to work at the lowest level possible that is available to us. Each touch has a unique identification (ID) that is given at the moment of TOUCHDOWN, to be used during the TOUCHMOVE, and to end when TOUCHUP has been activated. The ID gives us a very nice way to keep track of a trace, which is defined as the path of the touch from TOUCHDOWN to TOUCHUP.

33.3.1.3 Visual Display

As a test case, we have created a world with 64 by 64 by 64 spheres, in a cubic arrangement, drawn in perspective mode, where each sphere has a different color. This allows the user to test the 3D navigation provided by our gestures while having a visual feedback. It is important to note that we colored the spheres in an ordered fashion using the lowest RGB values in one corner and the highest values in the opposite corner of the cube of spheres.

33.3.2 Gesture Recognition

We decided to perform our own gesture recognition as opposed to using language oriented libraries [35]. The reason is that this gives us an ability to improve in current techniques, specially when performance is required. This also gives us the power to choose the best recognition techniques to use them in combination, while assuring that users do not notice performance degradation. Finally, we believe that working at a low level gives us control of the entire interaction environment. This will keep the process simple without using obfuscated libraries, that provide additional functionalities that may slow down the user experience.

For this particular study, we decided to use *ad hoc* recognition because of the few gestures we implemented. For example, for the swipe gestures shown in Fig. 33.2b, c, we determined if a given set of N points of the total M points from a given trace (path) increased either in x and/or y axes. If the value increases, then this qualifies as a swipe gesture.

However, this *ad hoc* method will be replaced with a template based matching [25–27], in particular an adaptation of $N [26] algorithm. The reason that this was not selected at this point, is that the $N algorithm will always yield a gesture. This means that if a user performs an unknown gesture, the algorithm will still classify the closest one. Thus, yielding an incorrect interaction. In future work, we will try to find a proper threshold or a modification to the $N algorithm to remove this constraint.

33.3.3 Gestures Interaction

We have decided to develop separate gestures for translation and rotation to understand what combinations are more efficient for the user for 3D navigation. This means that when rotating, the user will be rotating by a specific axis and translations will be performed using two axes. We also decided to provide simple gestures for our initial design to see the interaction of the users. Once we have collected more data about the interaction, we can create more complex gestures, if needed.

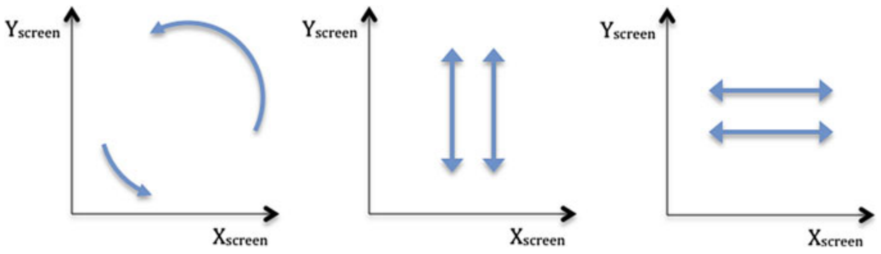

Fig. 33.2 Multi-touch rotation gestures. **a** About 3D x-axis; **b** About 3D z-axis; **c** About 3D y-axis

33.3.3.1 Translation

In order to translate the camera, we decided to combine the X- and Y-axes and leave Z by itself. The algorithm for the Y-axis is similar to Algorithm 1, replacing the variable X with the variable Y. The algorithm for the Z-axis is similar to the Y-axis with the exception that it uses 3 touches. In general, the user can perform simultaneous translation by the x and y axes using one finger or translate by the z-axis using three fingers. All the movements can be executed with a single hand.

33.3.3.2 Rotations

To address rotations, we have to think of rotation about x, y and z independently, given that is our belief that separating the rotation will demand a lower cognitive load from the user. This expectation is also supported by [9, 11, 12]. In addition, all the rotations are designed to use only one hand, which is preferable, as demonstrated in [9]. To keep the constraint of using only one hand, the algorithm checks that the touches are within a cluster.

The gesture for rotation about x, as shown in Fig. 33.3, merits to be described in more detail because the other two rotations about y and z use very similar algorithms to those already described for the translations. The only difference is that y and z rotations require two touches each. The gesture for rotation about x begins with T_1 and T_2, which form an angle α with the horizontal axis. The user's final state is represented with T_1' and T_2', forming an α' angle with the horizontal. Then, the difference between α' and α gives the rotation angle to be applied, about x.

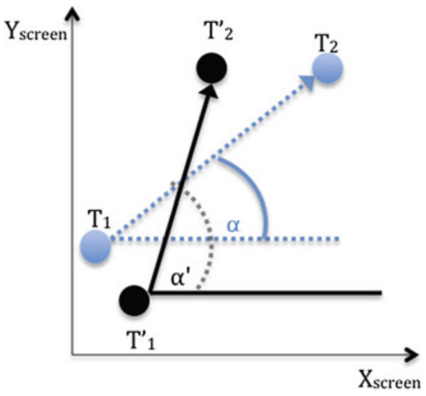

Fig. 33.3 Rotation about the 3D x-axis (see Fig. 33.1)

33.4 Discussion

33.4.1 Gestures

We believe that the set of gestures that we are proposing based on the literature reviewed in the background section and our own preliminary testing will give a starting point to find the most natural gestures to interact in a 3D world using a multi-touch device. Even after finding the most natural gestures for 3D navigation, one will have to compare with other devices such as the ones found in Bowman et al. [16]. As we will outline in the next section, we suggest to make the comparison with a 3D mouse [36].

The first question asked in the introduction was: What is the most appropriate mapping between the 2D interface surface and the 3D world? We have proposed a simple solution to the problem, through the set of gestures described above. Defining and implementing the most natural mapping between this 2D multi-touch interface and the 3D world may still require additional work, but the concepts advanced in this paper may provide an interesting direction towards the solution of this ongoing challenge.

The other question asked in the introduction was: Can previous techniques used with other devices (e.g., Joystick, keyboard and mouse) be used in 3D navigation? We propose that the answer is yes. We can build upon existing work that was developed for the mouse or other interfaces, adapting it for use with multi-touch displays whenever possible. An example of this is The Virtual Sphere [2]. We could take The Virtual Sphere and create a similar device for use with multiple fingers to allow a pure 3D rotation and translation, even emulating a 3D mouse [36]. However, those considerations would be outside the scope of this paper.

In general, we find that multi-touch displays can work efficiently for achieving a more natural user 3D interaction and 3D manipulation.

33.4.2 Proposed Evaluation Technique

The considerations presented above inform our current process of planning the experimental protocol and data analysis methods we will use for evaluating our approach. To answer our research questions and test the proposed gestures, we will recruit at least 30 subjects from the college student population at our university. The reason for our choice of target population is that we believe that all students will have a good grasp of basic mouse interaction, which will facilitate the completion of the experimental tasks by the subjects.

The actual experiment, after allowing the user to become familiarized with the interface, will consist of a set of tasks to test translation and rotation gestures (independently) using our 3M 22″ Multi Touch Monitor (Model M2256PW) and the 3D mouse made by 3DConnexion [36]. For each of the tasks, we will measure the time of execution to complete the task, and the accuracy of the movement. For the completion time, we will use an external game controller to start and stop the time, and for the accuracy of the movement, we will automatically record the initial and final positions. In addition to the automated recording of performance data, we will ask the subjects to complete a short usability questionnaire [37].

33.5 Conclusion

Recently, multi-touch displays have become more widely available and more affordable. Accordingly, the search for protocols that will simplify the use of these devices for interaction in 3D data environments has increased in importance. In this paper we have outlined some of the most valuable previous contributions to this area of research, highlighting some of the key past developments that have emerged in the 3D-interaction community. This review of pertinent literature provides a context for the presentation of the core elements of the solution we propose for the interaction in 3D environments through a multi-touch display.

Specifically, we proposed a set of multi-touch gestures that can be used to command translations and rotations in 3 axes, within a 3D environment. Our proposed solution has been implemented using a 3M M2256PW 22″ Multi-Touch Monitor as the interaction device. This paper explained the proposed gestures and described how these gestures are to be captured using the information provided by the device. In our definition of the proposed multi-touch gesture set we have established independent gestures for each type of translation and also for each type of rotation. We decided to proceed in this way so that we can study how users prefer to combine or concatenate these elementary gestures.

The next step in the development of our approach is to evaluate its efficiency in a comparative study involving other 3D interaction mechanisms, such as a 3D mouse. The ongoing process of planning the experiments for evaluation takes into account the nature of the devices and general principles of design of experiments, in an effort to minimize the presence of confounding effects, such as subject fatigue, etc. Our experiments may lead us to define alternative gestures to allow more innovative means of interaction.

Acknowledgments This work was sponsored by NSF grants HRD-0833093, and CNS-0959985. Mr. Francisco Ortega is the recipient of a GAANN fellowship, from the US Department of Education, at Florida International University.

References

1. Nielson G, Olsen D Jr (1987) Direct manipulation techniques for 3D objects using 2D locator devices. In: Proceedings of the 1986 workshop on Interactive 3D graphics, pp 175–182
2. Chen M, Mountford S, Sellen A (1988) A study in interactive 3-D rotation using 2-D control devices. ACM SIGGRAPH Comput Graph 22(4):129
3. Jacob R, Girouard A, Hirshfield L, Horn MS, Shaer O, Solovey ET, Zigelbaum J (2008) Reality-based interaction: a framework for post-WIMP interfaces. In: Proceeding of the twenty-sixth annual SIGCHI conference on human factors in computing systems (CHI '08), pp 201–210
4. Wang F, Cao X, Ren X, Irani P (2009) Detecting and leveraging finger orientation for interaction with direct-touch surfaces. In: Proceedings of the 22nd annual ACM symposium on user interface software and technology, pp 23–32
5. Wang F, Ren X (2009) Empirical evaluation for finger input properties in multi-touch interaction. In: Proceedings of the 27th international conference on human factors. ACM, Boston
6. Wilson A, Izadi S, Hilliges O, Garcia-Mendoza A, Kirk D (2008) Bringing physics to the surface. In: Proceedings of the 21st annual ACM symposium on user interface software and technology, pp 67–76
7. Cao X, Wilson A, Balakrishnan R, Hinckley K, Hudson S (2008) ShapeTouch: leveraging contact shape on interactive surfaces. In: TABLETOP 2008. 3rd IEEE international workshop on horizontal interactive human computer system 2008, pp 129–136
8. Benko H, Wilson A, Baudisch P (2006) Precise selection techniques for multi-touch screens. In: Proceedings of the SIGCHI conference on human factors in computing systems (CHI '06), pp 1263–1272
9. Nacenta MA, Baudisch P, Benko H, Wilson A (2009) Separability of spatial manipulations in multi-touch interfaces. In: GI '09 proceedings of graphics interface 2009. Canadian Information Processing Society, Toronto, May 2009
10. Hancock M, Carpendale S, Cockburn A (2007) Shallow-depth 3D interaction: design and evaluation of one-, two-and three-touch techniques. In: Proceedings of the SIGCHI conference on human factors in computing systems, p 1156
11. Kin K, Agrawala M, DeRose T (2009) Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. Canadian Information Processing Society, Toronto, May 2009
12. Moscovich T, Hughes J (2008) Indirect mappings of multi-touch input using one and two hands. In: Proceeding of the twenty-sixth annual SIGCHI conference on human factors in computing systems (CHI '08), pp 1275–1284
13. Glassner AS (1993) Graphics gems. Morgan Kaufmann, San Francisco
14. Arvo J (1994) Graphics gems II. Morgan Kaufmann, San Francisco
15. Heckbert PS (1994) Graphics gems IV. Morgan Kaufmann, San Francisco
16. Bowman DA (2005) 3D user interfaces: theory and practice. Addison-Wesley, Boston, p 478

17. Sezgin T, Davis R (2005) HMM-based efficient sketch recognition. In: Proceedings of the 10th international conference on intelligent user interfaces (IUI '05)
18. Hong P, Huang T (2000) Constructing finite state machines for fast gesture recognition. In: 15th international conference on pattern recognition (ICPR'00), vol 3, p 3695
19. Hong P, Huang T, Turk M (2000) Gesture modeling and recognition using finite state machines. In: IEEE conference on face and gesture recognition, Mar 2000
20. Pittman J (1991) Recognizing handwritten text. In: Human factors in computing systems: reaching through technology (CHI '91), New York, pp 271–275
21. Rubine D (1991) Specifying gestures by example. ACM SIGGRAPH Comput Graph 25(4):329–337
22. MacLean S, Labahn G (2010) Elastic matching in linear time and constant space. In: International workshop on document analysis systems 2010 (DAS '10)
23. Kara L, Stahovich T (2005) An image-based, trainable symbol recognizer for hand-drawn sketches. Comput Graph 29(4):501–517
24. Notowidigdo M, Miller R (2004) Off-line sketch interpretation. In: AAAI fall symposium, pp 120–126
25. Wobbrock J, Wilson A (2007) 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 (UIST '07)
26. Anthony L, Wobbrock J (2010) A lightweight multistroke recognizer for user interface prototypes. In: Proceedings of graphics interface 2010 (GI'10), Toronto
27. Li Y (2010) Protractor: a fast and accurate gesture recognizer. In: Proceedings of the 28th international conference on human factors in computing systems (CHI '10), New York
28. Johnson G, Gross M, Hong J (2009) Computational support for sketching in design: a review. Found Trends Human–Comput Inter 2: 1–93
29. Press WH, Flannery BP, Teukolsky SA, Vetterling WT (2007) Numerical recipes, 3rd edn. The art of scientific computing. Cambridge University Press, Hong Kong
30. Kruger R, Carpendale S, Scott S, Tang A (2005) Fluid integration of rotation and translation. In: Proceedings of the SIGCHI conference on human factors in computing systems, pp 601–610
31. Reisman J, Davidson P, Han J (2009) A screen-space formulation for 2D and 3D direct manipulation. In: Proceedings of the 22nd annual ACM symposium on User interface software and technology, pp 69–78
32. Wright RS, Haemel N, Sellers G, Lipchak B (2010) OpenGL superbible. Comprehensive tutorial and reference. Addison-Wesley, Reading
33. Han J, Kim J (2011) 3D graphics for game programming. Chapman and Hall, London
34. Kiriaty Y, Moroney L, Goldshtein S, Fliess A (2009) Introducing windows 7 for developers. Microsoft Press, Redmond
35. Laufs U, Ruff C, Zibuschka J (2010) MT4j-a cross-platform multi-touch development framework. In: ACM EICS 2010, workshop: engineering patterns for multi-touch interfaces, pp 52–57
36. O'Brien T, Keefe D, Laidlaw D (2008) A case study in using gestures and bimanual interaction to extend a high-DOF input device. In: Proceedings of the 2008 symposium on interactive 3D graphics and games (I3D '08), New York
37. Lazar J, Jinjuan Heidi Feng D, Harry Hochheiser D (2010) Research methods in human–computer interaction. Wiley, New York