Intuitive Placement of Objects in Web-Based CAD Environments

Andres Felipe Kordek[1] and Arjan Kuijper[1,2(✉)]

[1] Technische Universität Darmstadt, Darmstadt, Germany
[2] Fraunhofer IGD, Darmstadt, Germany
arjan.kuijper@igd.fraunhofer

Abstract. We develop a Computer Aided Design (CAD) editor using an open source library, with the aim to minimize the cost in the future, as compared to the development of conventional editors, and to accelerate the expansion by means of standardized languages making the development easier. We focus on snapping, a very important area in computer graphics and without a CAD application inconceivable. CAD applications offer the possibility of snappings to allow the developers an intuitive interaction with the objects in 2D or 3D space. Snapping allows by using constraints the merging of multiple objects into a new object. Two possible approaches for implementing snappings are addressed and presented. Advantages and disadvantages are discussed via a user study.

1 Introduction and Motivation

The use of current web technology has great potential for improving collaborative design and construction processes. Apart from a Web browser, no special software is needed, so that user groups with various software platforms and devices can participate in the design process immediately. With the number of untrained CAD users also the demands on the intuitive usability of the 3D application increase. In particular, the millimeter-accurate positioning of an object to the desired location in three-dimensional space is a challenge [3]. Although the desired result can be achieved using manual position information, this takes quite a bit of time. It is possible in the positioning of the objects to use so-called constraints, i.e. context-dependent rules. These are defined per object and guide the user in the ideal case quickly to the desired result. However, constraints can even interfere with the usability of the application, such as in cases where a proposed solution does not quickly lead to the desired result.

In this work two constraint-based models for object placement are implemented and compared. The focus is on the intuitive usability, which is evaluated in a user study. The first model is already in use for many years (via a command line interface) at an industrial partner. Users specify which point of an object (selected from a predefined set of options) is to be connected with what point of a neighboring object. The application then automatically calculates the desired position and orientation of the object. Instead of a command-line interface a simple list within the graphical user interface for selecting the points is

C. Stephanidis (Ed.): HCII 2015 Posters, Part I, CCIS 528, pp. 414–420, 2015.
DOI: 10.1007/978-3-319-21380-4_70

developed in this work. The second model is known from similar applications: The user moves an object with the mouse. As soon as the object goes to the proximity of another object, the application automatically suggests a positioning and orientation (so-called snapping [2,3,7]) of the object which then satisfies predetermined constraints.

The editor is a web-based application, developed as open source with standard technologies. It should serve as a tool for the design of digital factories [12] and makes components available, which can be combined to factory elements. Snapping is in CAD editors a very important tool that can built large modules with the basic components. The editor that comes in question here, provides two components: a tank and a pipe. The idea in the implementation is to expand the components with snap points and normal vectors. The snap points are the connection points of the object and the normal vectors determine the direction of the point. So it can be determined, which snap points fit together and can thus be connected using defined rules. The normals help in deciding the orientation of an object in relation to another.

The objectives of the industrial partner include the use of standard technologies, open source frameworks, and the development of a web-based CAD editor [1,9,10]. The editor is invoked by a web browser and gives the feeling of a local application. In order to ensure the requirements the following technologies are used: HTML, CSS, JSON, JavaScript, jQuery, X3D, and X3DOM [5,6,12].

2 Snapping in Computer Graphics

We present two approaches that are specially adapted to the requirements of CAD editors. It is analyzed in what way both approaches can interact with each other and what advantages and disadvantages they have. Since the early days of computer graphics new implementations are presented again and again to facilitate the user interaction. Especially with the development of CAD and graphics solutions technologies as snapping have been introduced and constantly improved. In the design of new approaches user perception is important. An example is the perception of depth in three-dimensional space, or the perception of the center of a non-symmetric element. Equally important is the user interaction that takes place with the help of a mouse and the keyboard or other input elements. A user study [7] examined the perception of asymmetric objects to a user group. This study shows the importance of considering these characteristics when creating new approaches and algorithms.

In computer graphics, the problem of aligning objects in three-dimensional space is dedicated a lot of time. Many operations must be considered and the selection of the correct control for the elements must be carefully considered. In addition, the necessary requirements must be fulfilled [3]. An approach that is considering this question, is snap-dragging [3,4]. With the help of three interactive techniques – *gravity*, *alignment*, and *transform* – the control and the design of elements is described in space. Gravity locates by means of an algorithm the point in three-dimensional space, defining the depth of a 3D scene imaged on

a two-dimensional screen. The alignment-objects are lines, planes and spheres, with special properties that allow the modeling of 3D scenes. Transformations such as scaling, rotation and translation operate with the help of specific constraints on the existing alignment-Objects.

A very interesting approach, which addresses a different problem with snapping, is *Snap-and-Go* [2]. It demonstrates that it is not always desirable that an object is attracted to one or more snap points when dragged with the mouse. As a solution, the functionality of the snappings was temporarily disabled. This decision is troublesome for each user, if it is assumed that snapping is meant to facilitate usability. This property is not given in such a case. The approach *Snap-and-Go* describes how disturbing snap points are handled without disabling snapping. This allows a quick and intuitive orientation of objects and increases the usability of an editor. In the design of software projects, the user is very prominent in the foreground. He sets the destination, where the development of a product should result in. User tests are very important [7], they give insight into the perception and usability [8] of a product. Both our implementations of snappins are therefore evaluated in a user study.

3 Constraints Modeling Concepts

Constraints in computer graphics are fixed terms which describe the motion behavior and properties of an object. In this work, it is basically the behavior of objects on one another. The snapping task is to find ways that objects can be connected to each other in 3D space. To achieve this, properties must be calculated as the distance and position of objects to another. For example, features such as scaling and rotation with respect to the local and global axis system are of importance. These geometric relationships between the objects form the basis of both constraints.

Semi-Automatic Snapping: The first model illustrates and explains how snapping between objects in 3D space is performed using the context menu. The original editor has a similar process. The commands are transmitted to a command interface and then performed snapping. The idea behind the implementation using a context menu is to build a user-friendly and intuitive environment. Each element in space should have the property to trap the click events of the mouse and interpret them. Depending on given conditions, a context menu appears that lists the existing snap points elements. The following requirements are tested before the mouse event can be evaluated: (i) at least two objects must be available in 3D space; (ii) in the top bar snapping with context window must be turned on, and (iii) the snap points must be generated and displayed. Upon successful verification, a context window is shown next to the object, see Fig. 1, left. A special feature of this context window is the nature of their implementation. In case of repeated click event, the old window is deleted and a new one opened. For each item that is clicked, a new context window is generated and displayed increasing flexibility and supporting the extensibility of the snappings for future projects. Always two points can be connected. From the context window of the

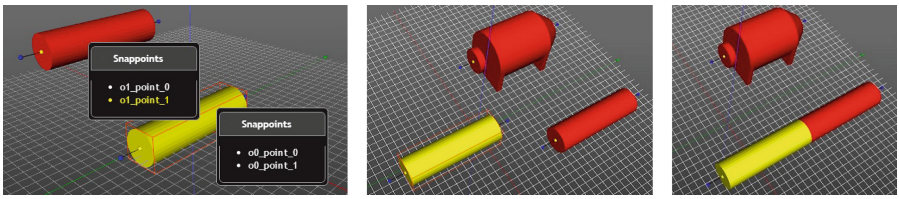

Fig. 1. Example objects and computed connection. Left: Semi-automatic with context menu. Middle: Found connections in the full automatic snapping. Right: Proposed connection.

first element a starting point is selected and from the context window of another element the target point. The object with the start point is shifted to the object with the target point and positioned so that both snap points are connected. To give the user an intuitive feeling, fixed properties of each element are present. The selected item is always highlighted in yellow. As another property snappoints change color when the user goes over a point in the list item in the context menu with the mouse. When an item is selected in the context menu of the list the entry turns blue, so the user always knows which item he has selected.

Fully Automatic Snapping: In the second model the intuitive concept Snapping is implemented using the mouse.

In the development and implementation of fully automatic snappings within the Web-based CAD editor, the consideration of the right approach and the more accurate representation of the problem played a very important role. Since the CAD editor has been implemented using standardized web technologies, the decision of the development language for JavaScript was determined. The approach was taken in the field of software development using design patterns. The requirements provided by the task have lead to the implementation of the observer pattern for solving the problem.

The task is to implement the snappings with intuitive behavior. It should be ensured that the distance between the objects is considered. The first variant implements snapping with the aid of the so-called context menus. The user can call it up by clicking with the right mouse button on the element. The second implementation enables snapping out of context menus but by using the mouse only. An item is clicked with the mouse and dragged to the next element. The distance between the elements is always calculated. If two snap points are nearby, they attract and connect themselves, see Fig. 1, middle and right.

4 Results

A user evaluation was carried out test the usability of the implemented snapping methods in the CAD editor [11]. A test environment was developed for the test persons. After the test the respondents completed a questionnaire. The choice of subjects is tied to the question for which groups the CAD editor was

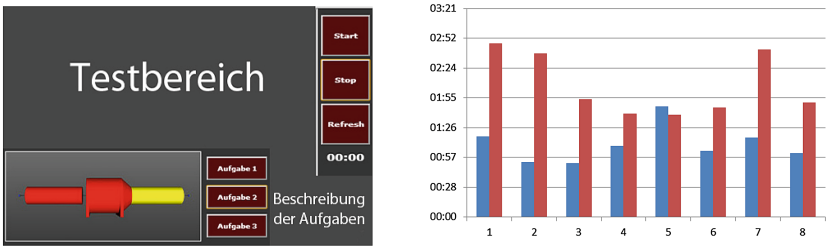

Fig. 2. Left: Testing scenarios. Right: Total timing per question person with left bar full automatic, right bar semi-automatic.

basically developed. The choice fell on eight subjects. All eight subjects come from the department of computer science and have already prior knowledge of graphic editors. The subjects are all between 27 years and 32 years old. We created a total of three test tasks for the volunteers. Each task has about the same level of difficulty. The subject had to combine several components so that the arrangement of them corresponded to the given device. Each of the three test tasks had to be performed both with the fully automatic snapping and with the semi-automatic snapping. Figure 2, left, shows the arrangement of the test environment. Top right the time is measured used for the successful completion of a task. Bottom left, the subject can choose a task and gets for each task a new image displayed that had to be assembled. When preparing the questionnaire, questions were specifically selected asking about the usability of both implementations of the snappings. The aim was to draw conclusions that are important in the decision of improvements for both methods. The respondent has five different possible answers to each question. The weighting of each possible response is 1 for strongly agree and 5 for statement is not true.

1. By using the fully automatic snappings the intuitive interaction with the components is possible ($\mu = 1.250$).
2. By using the semi-automatic snappings the intuitive interaction with the components is possible ($\mu = 2.875$).
3. I quickly learned how to use the fully automatic snapping ($\mu = 1.125$).
4. I quickly learned how to use the semi-automatic snapping ($\mu = 2.000$).
5. I get along very well with the fully automatic snapping ($\mu = 1.375$).
6. I get along very well with the semi-automatic snapping ($\mu = 2.625$).
7. In the end, I could do all tasks using the fully automatic snappings easier and faster than using the semi-automatic snappings ($\mu = 1.875$).
8. Displaying a context menu contributes to the clarity ($\mu = 2.875$).
9. Semi-automatic snapping displaying context menus is more user friendly than the fully automatic snapping out of context menus ($\mu = 3.625$).
10. With the fully automatic snapping the task needs fewer steps than the semi-automatic snapping ($\mu = 1.375$).

Concluding, the implementation of fully automated snappings was favorably to the subjects. Both Question 1, 3, 5, 7 and 10, rating the statements of the

fully automatic snapping , were graded with a 1 at least 5 of 8 subjects. The statements on the semi-automatic snapping question (Question 2, 4, 6, 8 and 9), show that the majority of subjects was not in favor of it.

Figure 2 right combines the time measurements for both methods. We see that the satisfaction of the subjects with the fully automatic snapping is in line with the measured time.

5 Conclusions

In the context of this thesis, two methods were developed and presented to the objects in three-dimensional space to connect with each other, so-called snapping. Both methods are called semi-automatic and fully automatic snapping. They have different approaches and work according to certain specifications. Here, a CAD editor originated served as the setting for the implementation of both methods. Simultaneously, a user evaluation was carried out. This new insight into the usability of both Snapping obtained implementations. Thus, there was clear from the evaluation that the subjects with the fully automatic snapping basically came efficiently than the semi-automatic snapping. Several reasons were mentioned. First and foremost, however, was the ease of use and intuitive behavior in the foreground. Bottom line is that the use of fully automated Snappings contributes to the clarity and is easy to use. This in turn leads to a better usability and is thus preferable to the semi-automatic snapping.

References

1. Aderhold, A., Wilkosinska, K., Corsini, M., Jung, Y., Graf, H., Kuijper, A.: The common implementation framework as service – towards novel applications for streamlined presentation of 3d content on the web. In: Marcus, A. (ed.) DUXU 2014, Part II. LNCS, vol. 8518, pp. 3–14. Springer, Heidelberg (2014)
2. Baudisch, P., Cutrell, E., Hinckley, K., Eversole, A.: Snap-and-go: helping users align objects without the modality of traditional snapping. In: SIGCHI Conference on Human Factors in Computing Systems, pp. 301–310. ACM (2005)
3. Bier, E.A.: Snap-dragging in three dimensions. In: Proceedings of the 1990 Symposium on Interactive 3D Graphics, I3D '1990, pp. 193–204 (1990)
4. Bier, E.A., Stone, M.C.: Snap-dragging. In: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, pp. 233–240 (1986)
5. Eicke, T.N., Jung, Y., Kuijper, A.: Stable dynamic webshadows in the x3dom framework. Expert Syst. Appl. **42**(7), 3585–3609 (2015)
6. Engelke, T., Becker, M., Wuest, H., Keil, J., Kuijper, A.: MobileAR browser - a generic architecture for rapid AR-multi-level development. Expert Syst. Appl. **40**(7), 2704–2714 (2013)
7. Heo, S., Lee, Y.K., Yeom, J., Lee, G.: Design of a shape dependent snapping algorithm. In: CHI 2012 Extended Abstracts on Human Factors in Computing Systems, pp. 2207–2212 (2012)
8. Jokela, T., Iivari, N., Matero, J., Karukka, M.: The standard of user-centered design and the standard definition of usability: analyzing iso 13407 against iso 9241–11. In: Proceedings of the Latin American Conference on Human-computer Interaction, pp. 53–60 (2003)

9. Limper, M., Jung, Y., Behr, J., Sturm, T., Franke, T., Schwenk, K., Kuijper, A.: Fast, progressive loading of binary-encoded declarative-3d web content. IEEE Comput. Graph. Appl. **33**(5), 26–36 (2013)
10. Mouton, C., Parfouru, S., Jeulin, C., Dutertre, C., Goblet, J., Paviot, T., Lamouri, S., Limper, M., Stein, C., Behr, J., Jung, Y.: Enhancing the plant layout design process using X3DOM and a scalable web3d service architecture. In: The 19th International Conference on Web3D Technology, Web3D14, pp. 125–132 (2014)
11. Nazemi, K., Stab, C., Kuijper, A.: A reference model for adaptive visualization systems. In: Jacko, J.A. (ed.) Human-Computer Interaction, Part I, HCII 2011. LNCS, vol. 6761, pp. 480–489. Springer, Heidelberg (2011)
12. Stein, C., Limper, M., Kuijper, A.: Spatial data structures to accelerate the visibility determination for large model visualization on the web. In: The 19th International Conference on Web3D Technology, Web3D14, pp. 53–61 (2014)