Paper and Pen: A 3D Sketching System

Cansın Yıldız and Tolga Çapın

Abstract This paper proposes a method that resembles a natural pen and paper interface to create curve based 3D sketches. The system is particularly useful for representing initial 3D design ideas without much effort. Users interact with the system by the help of a pressure sensitive *pen* tablet. The input strokes of the users are projected onto a drawing plane, which serves as a *paper* that they can place anywhere in the 3D scene. The resulting 3D sketch is visualized emphasizing depth perception. Our evaluation involving several naive users suggest that the system is suitable for a broad range of users to easily express their ideas in 3D. We further analyze the system with the help of an architect to demonstrate the expressive capabilities.

1 Introduction

3D modeling starts with rough sketching of ideas. The latest efforts in research on the field have focused on bringing the natural pen and paper interface to 3D modeling world. The complicated and hard-to-learn nature of current *WIMP* (windows, icon, pointer, menu) based 3D modeling tools is the reason for the search of a better interface. Several authors has already recognized the importance of this problem [18]. In this paper, we present a method that tries to mimic the natural interface of pen and paper for creating 3D sketches that can be used an easier way to represent ideas in 3D without much effort. The system is designed to be as minimalistic and simple as possible, since it targets a broad range of users, from expert designers to

C. Yıldız (✉) · T. Çapın
Department of Computer Engineering, Bilkent University,
06800 Bilkent, Ankara, Turkey
e-mail: cansin@cs.bilkent.edu.tr
URL: http://cs.bilkent.edu.tr

T. Çapın
e-mail: tcapin@cs.bilkent.edu.tr

naive users. We test whether we are able to achieve this or not, through several user tests (Sect. 5).

Our system is based on the very idea of *curves*, rather than 3D solid objects. Concern of creating surfaces not in mind, it is much easier to develop complicated 3D scenes and objects. Although there are several other examples of a similar 3D sketching interface, our contribution to the field is to explain an easy to use 3D sketching tool, that is designed with *less is more* [17] thought in mind. Several different sketching tools also developed during implementation of the system.

2 Related Work

Creating 3D objects and curves from 2D user interfaces has been studied for a long time [7, 13, 21]. There are several recent research on the subject that tries to enrich an already existing 3D scene, either by annotating the scene or augmenting the 3D object itself [9, 14, 15]. Bourguignon et al. [9] created a system that can be used both annotating a 3D object or creating an artistic illustration that can be represented from different viewpoints. Although the resulting scenes are pleasingly beautiful, they are not truly 3D. The system mimics a 3D perspective by manipulating the curves' render mechanism according to the angle they make with the viewport. At Kara et al.'s [14, 15] work, a true 3D object is created by augmenting a simpler pre-loaded 3D template of the target object. Simply, if user wants to create a fancy chair, a simpler chair model is loaded beforehand, which the user can edit with a sketch interface.

Bae et al.'s *ILoveSketch* [3], and later extended version *EverybodyLovesSketch* [4], rely on the idea of creating curve-based 3D scenes rather than traditional plane-based 3D models as we do. Their approach uses several different drawing techniques and navigation tools that a user can select from. Although it is easy to learn the entry point to 3D sketching ideas such as *orthographic plane sketching* and *single-view symmetric curve*, it takes some time to learn how to use the system in depth, as they noted [4]. In our system, we have chosen to use only a single way of drawing and navigating (as explained in Sect. 3.1), which makes it much easier to learn.

3 Overview of the System

Users interact with our system using a pressure sensitive pen tablet. In essence, users' pen gestures are captured as a time sequenced tablet coordinates and interpreted. The device has several buttons on the tablet that are used for some basic non gestural abilities like undo, redo, toggle symmetry. The pen also has two buttons, and an eraser at back, that are used as toggles between our gesture modes as detailed in the following Sect. 3.1.

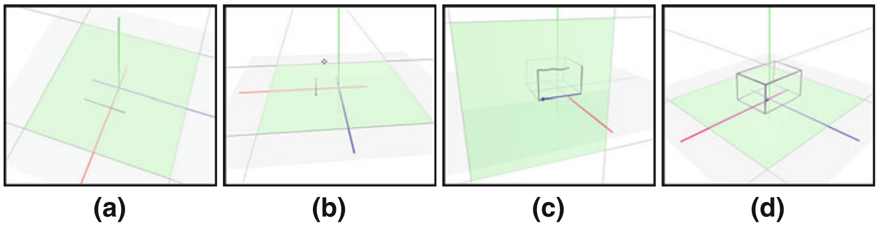

Fig. 1 Overview of the system. **a** User adjusts the drawing plane; and draws the curve using pen tablet. **b** The camera position can be changed. **c** The process is repeated until the desired 3D sketch is formed. **d** Final result

Figure 1 illustrates the overall usage of the system. To be able to draw a curve, the user firsts adjust the *drawing plane* as explained in Sect. 3.1. Any drawing gesture that is made by the user will be reflected on this surface. Once the drawing plane is adjusted, the user can draw a curve with a simple pen gesture on the tablet. The input curve will then be re-sampled and smoothed using the algorithms described at Sect. 4.1. During this process, the user can adjust camera position as well, using the same pen tablet device, if necessary. The user can repeat these steps to complete the 3D object.

3.1 Gesture Modes

The pen tablet acts like a *modal interface* for users, allowing it to be used for several different tasks. The user can switch between different modes by holding down the buttons on the pen. On a regular session with the system, one will use the pen for camera adjustment, plane selection, drawing and erasing.

- **Camera Adjustment** When the pen is in this mode, every movement user does will be mapped to an invisible *Two-Axis Valuator Trackball* [11]. The horizontal pen movement is mapped to a rotation about the up-vector, whereas a vertical pen movement is mapped to a rotation about the vector perpendicular to view and up. As Bade et al. suggested [2], Two-Axis Valuator Trackball is among “the best 3D rotation technique” among several rotational widgets.
- **Plane Selection** When the user draws curves with the tablet, these curves should be reflected onto a virtual surface at 3D scene. To enable this effect, the user should select a *drawing plane* beforehand. In our system, there are only two distinct ways of selecting the drawing plane.
  - In first approach, the user takes assistance from coordinate system lines and current curves on the scene. By selecting any of these curves, the user changes the drawing surface as the plane that selected curve lies on. Further flexibility is enabled with the help of *toggle plane* button on the tablet. Once that button is pushed, the drawing surface will be changed to one of the planes that forms

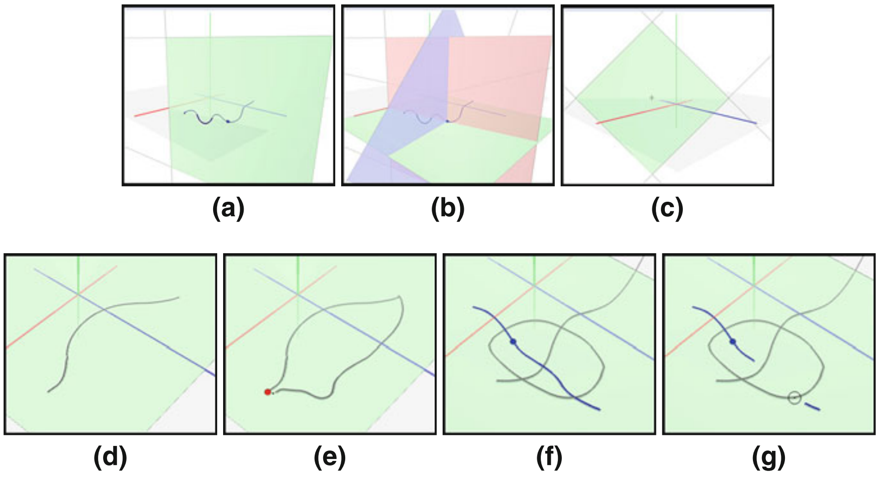

Fig. 2 **a, b** Plane selection with a Cartesian system or (**c**) extruding a picking ray. **d** Drawing gesture with (**e**) snap points. **f, g** Erasing

a Cartesian system with the drawing plane and tangential to the curve at the selected point (Fig. 2a, b).

– To support even more flexibility, we realized a second approach to plane selection. In this method, the user can adjust the drawing surface to a plane that is parallel to the current near plane of the scene's viewport, and x distant from that near plane, where that x is determined by the current pressure on the pen (Fig. 2c).

- **Drawing** The main functionality of the system, is drawing curves (Fig. 2d). In this mode, the user can simply draw several curves using pen tablet. The time sequenced (x, y) data that is collected from the pen tablet is then projected to the current drawing plane. After the projection is performed, several re-sampling and smoothing algorithms are used to ensure a plausible curve shape, as detailed in Sect. 4. Finally, a B-Spline curve is fitted to the stroke data. While in the drawing mode, the user can take advantage of *snap points* that will appear at the start and end points of existing curves (Fig. 2e). These snap points make it even easier to draw closed or connected shapes.
- **Erasing** A paper and pen system cannot be imagined without an eraser. The user can simply turn over his pen device to switch to the eraser mode. Once this is done, the cursor on the screen will get larger to mimic an eraser functionality. Since in a crowded scene, there will be several curves that will lie under eraser's cursor, it will be harder to erase a specific curve's segment. Therefore, erasing can only be performed on the current *selected* curve (Fig. 2f, g).

As mentioned, there are also several buttons on the tablet, that can be used to achieve some misc. operations. When symmetry is toggled, any gesture that's performed with the pen will also be reflected to the symmetry of that gesture. Symmetry is important to product design, since people prefer objects with symmetry, unity and harmony [8].

To prevent errors that the users might make, the system changes the pen's cursor's image to reflect the current gesture mode [1]. For instance, it's a single dot for drawing, a bigger circle for erasing, a cross-hair for plane selection etc. Similarly, our system also supports undo/redo actions using the tablet buttons as well. This functionality is really essential for basic error recovery. As can be seen in Sect. 5, undo is widely used among our users.

4 Implementation Details

There is a common pipeline [18] for *sketch based interfaces*, which our system also follows. The first step is to acquire input from the user, by means of an input device, a pen tablet in our case. That step is followed by sketch filtering, where the data is re-sampled and smoothed. Finally, the sketch is interpreted appropriately.

4.1 Sketch Acquisition and Filtering

Obtaining a sketch from the user is the first step a sketch based interface should perform. Our system collects free hand sketches from the user using a pen tablet. A tablet display would be even a better choice, since the user will be able to see what he draws just at the drawing surface he is using.

It is important to perform filtering before storing a sketch to the system, since there will be some error caused by both user, and the input device itself [19]. Therefore, the input data should be interpreted knowing that it is imperfect. To overcome this imperfection, our system applies below approaches:

- **Re-sampling and Smoothing** The distance between data samples that are acquired from the pen tablet is not always the same (Fig. 3b). Therefore a re-sampling mechanism is needed to normalize distances (Fig. 3c). To further smooth out the given input, we use a local Gaussian filter (Fig. 3d) to any upcoming data point [20].
- **Fitting** After re-sampling and smoothing is performed, the resulting curve consists of hundreds of data points. To simplify this representation, we fit a curve onto these data points, using Reverse Chaikin Subdivision [6]. At every iteration of this algorithm, the data size halves. After appropriate number of iterations, these coarse points are used as control points for a B-Spline curve (Fig. 3e). Assuming fine points are denoted as $p_1, p_2, \ldots, p_n$, and coarse points are denoted as $c_1, c_2, \ldots, c_n$, a coarse point c_j can be computed as follows:

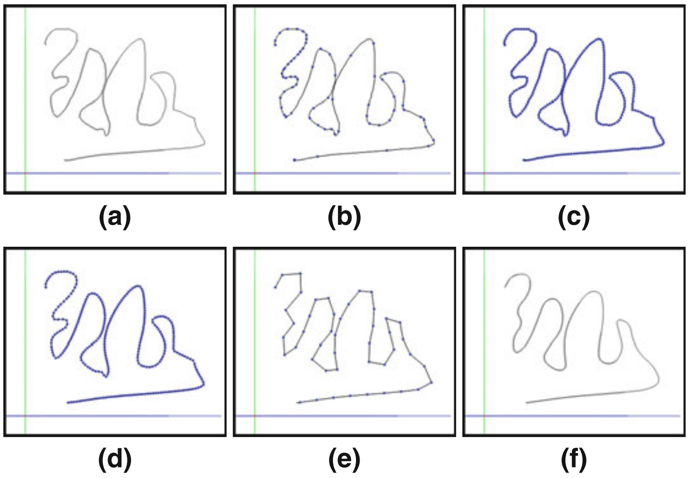

Fig. 3 **a** Initial user input. **b** Non-uniform distribution (706 points). **c** Re-sampled (2877 points). **d** Gaussian filtered. **e** Reverse Chaikin subdivided (47 points to *represent*). **f** Final B-spline curve (188 points to *render*)

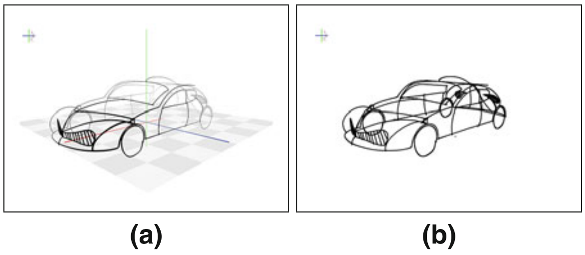

Fig. 4 Visualization at our system: **a** with depth cues; **b** without depth cues

$$c_j = -\frac{1}{4}p_{i-1} + \frac{3}{4}p_i + \frac{3}{4}p_{i+1} - \frac{1}{4}p_{i+2} \qquad (1)$$

4.2 Visualization

Correct visualization of a scene is fairly important to make it easier for users to understand the 3D information behind the scene. In technical illustrations, there are three line conventions suggested by Martin [16]: use single line weight throughout the image; use heavy line weights for out edges, and parts with open space between them; or vary line weight to emphasize perspective (i.e. thicker is closer).

Since our concern is to emphasize 3D recognition as much as possible, we find third convention most suitable for the system. As can be seen at Fig. 4, by varying

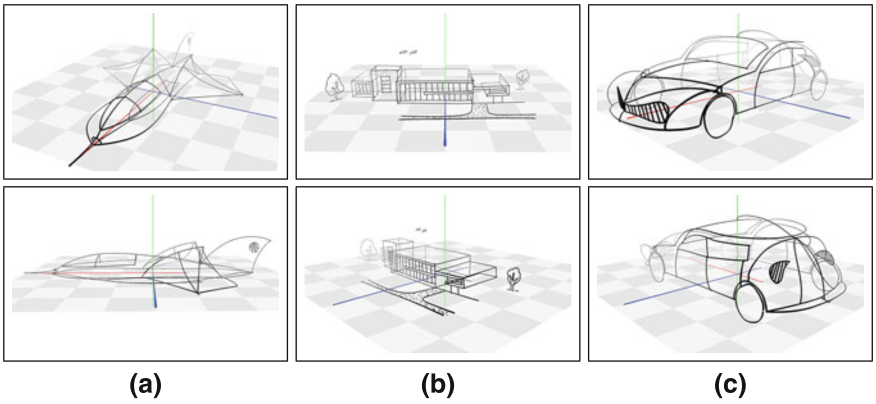

Fig. 5 Sample results. **a** A jet fighter. **b** A building complex. **c** A car

both line thickness and opacity with respect to z distance from the camera, our system makes it a lot easier to recognize the shape of 3D objects.

5 Results and Discussion

We invited an architect to perform a subjective expert evaluation. After an brief introductory explanation of the system for 15 min, the architect is left alone with the system for a full day. The resulting objects of the day can be seen at Fig. 5. The architect stated that he did like using the system, but he thinks such a system is more suitable for product design rather than architectural design. We agree on this comment, since our system tries to emphasize the power of free form curves, it is actually a bit harder to create regular shapes such as cubes and pyramids. One usability issue that we noted was that the architect preferred using undo function instead of erasing gesture most of the time. Only for some small adjustments, like shortening a curve which is a little too long, he used erasing.

We have also performed objective formal experiment to evaluate the usability of the system. We have selected twelve users that do not have prior experience with technical or artistic drawing, and pen tablets. In a standard test case, we introduced the system to each user briefly within five minutes. Then, we asked them to exactly copy the object they see on the scene. The test has twelve objects, some of which are 2D regular shapes, while others consist of 3D objects.

On average it took 67 s to draw a 2D object for all users, whereas it took 301 s for a 3D one. The slight complexity of 3D objects, and the need to adjust drawing plane several times, made 3D objects need more time to draw. We evaluate the resulting scenes with the goal objects using Modified Hausdorff Distance, as described in [12]. Evaluation suggests that the error for 3D objects is not that different from 2D object errors, and on average that error is 0.12 for 2D scenes, and 0.14 for 3D scenes. Given a common object had at most 8 length dimension, 1.62 % (0.13 over 8) is indeed not a significant error. Therefore, we can say that it was as easy to draw 3D objects as it is for 2D objects.

We also conduct a System Usability Scale Survey for each user, at the end of the test. System Usability Scale is a simple, ten-item Likert scale giving a global view of subjective assessments of usability [10]. Over twelve test users, our system got 83.75 out of 100 which can be referred as an "excellent" or "B" grade system, according to Bangor et al.'s work [5].

6 Conclusions

We have created a 3D sketching system that can be broadly used by any user, almost like a 3D *paint*. We did push the limits of the system by working with a professional architect to see what the system is capable of, whereas we also test the system with naive users with a more simplistic way. These evaluations show that our system is an easy to use, yet capable 3D curve sketching interface that requires little learning effort.

References

1. Andre, A., Degani, A.: Do you know what mode you're in? an analysis of mode error in everyday things. In: Mouloua, M., Koonce, J.M. (eds.) Human-Automation Interaction: Research and Practice. Lawrence Erlbaum, Mahwah (1997)
2. Bade, R., Ritter, F., Preim, B.: Usability comparison of mouse-based interaction techniques for predictable 3d rotation. In: Proceedings of Smart Graphics (2005)
3. Bae, S., Balakrishnan, R., Singh, K.: ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In: Proceedings of UIST '08 (2008)
4. Bae, S., Balakrishnan, R., Singh, K.: Everybodylovessketch: 3d sketching for a broader audience. In: Proceedings of UIST '09 (2009)
5. Bangor, A., Miller, J., et al.: Determining what individual sus scores mean: adding an adjective rating scale. J. Usability Stud. **4**(3), 114–123 (2009)
6. Bartels, R. Samavati, F.: Reversing subdivision rules: local linear conditions and observations on inner products. J. Comput. Appl. Math. **119**, 29–67 (2000)
7. Baudel, T.: A mark-based interaction paradigm for free-hand drawing. In: Proceedings of UIST '94 (1994)
8. Bloch, P.: Seeking the ideal form: product design and consumer response. J. Mark. **59**, 16–29 (1995)
9. Bourguignon, D., Cani, M., Drettakis, G.: Drawing for illustration and annotation in 3d. Comput. Graph. Forum **20**, 114–122 (2001)
10. Brooke, J.: Sus-a quick and dirty usability scale. In: Weerdmeester, B.A., McClelland, I.L. (eds.) Usability Evaluation in Industry. Taylor and Francis, London (1996)
11. Chen, M., Mountford, S.J., Sellen, A.: A study in interactive 3-d rotation using 2-d control devices. In: Proceedings of SIGGRAPH '88 (1988)

12. Dubuisson, M., Jain, A.: A modified hausdorff distance for object matching. In: Proceedings of the 12th IAPR (1994)
13. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: a sketching interface for 3d freeform design. In: Proceedings of SIGGRAPH '99 (1999)
14. Kara, L., Shimada, K. Construction and modification of 3d geometry using a sketch-based interface. In: Proceedings of SBIM 06 (2006)
15. Kara, L., Shimada, K.: Sketch-based 3d-shape creation for industrial styling design. IEEE Comput. Graph. Appl. **27**, 60–71 (2007)
16. Martin, J.: Technical Illustration: Materials, Methods and Techniques/Judy Martin. Child and Associates, Frenchs Forest (1989)
17. Nielsen, J.: Usability Engineering. Morgan Kaufmann, San Francisco (1994)
18. Olsen, L., Samavati, F., Sousa, M., Jorge, J.: Sketch-based modeling: a survey. Comput. Graph. **33**, 82–109 (2009)
19. Sezgin, T., Davis, R.: Scale-space based feature point detection for digital ink. In: ACM SIGGRAPH 2007 courses (2007)
20. Taubin, G. Curve and surface smoothing without shrinkage. In: Proceedings of the 5th ICCV (1995)
21. Zeleznik, R., Herndon, K., Hughes, J.: Sketch: an interface for sketching 3d scenes. In: Proceedings of SIGGRAPH '96 (1996)