Facetons: Face Primitives with Adaptive Bounds for Building 3D Architectural Models in Virtual Environment

Naoki Sasaki* Hsiang-Ting Chen† Daisuke Sakamoto‡ Takeo Igarashi§

JST ERATO Igarashi Design Interface Project / The University of Tokyo

Abstract

We present *faceton*, a geometric modeling primitive designed for building architectural models, using a six degrees of freedom (DoF) input device in a virtual environment (VE). A *faceton* is given as an oriented point floating in the air and defines a plane of infinite extent passing through the point. The polygonal mesh model is constructed by taking the intersection of the planes associated with the *facetons*. With the simple drag-and-drop and group interaction of *faceton*, users can easily create 3D architecture models in the VE. The *faceton* primitive and its interaction reduce the overhead associated with standard polygonal mesh modeling in VE, where users have to manually specify vertexes and edges which could be far away. The *faceton* representation is inspired by the research on boundary representations (B-rep) and constructive solid geometry (CSG), but it is driven by a novel adaptive bounding algorithm and is specifically designed for the 3D modeling activities in an immersive virtual environment.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric Algorithms

Keywords: geometric modeling, polygonal mesh, virtual reality

1 Introduction

IImmersive environments can be an ideal geometric modeling interface for architecture because they allow situated, body-scale interactions. The user can put up walls in front of them as if he or she were in the actual environment. An example of such a vision can be seen in a popular short film called World Builder [Branit 2007].

Previous research projects integrated computer-aided design (CAD) and CSG software frameworks in immersive environments [Butterworth et al. 1992, Fernando et al. 1999, Ma et al. 2004] nd explored the possibility of using different user interfaces for 3D modeling in VE, such as sculpting [Galyean and Hughes 1991], hand gestures [Tsang et al. 1998, Matsumiya et al. 2000], hand motions [Schkolne et al. 2001], speech and 6-DoF controllers [Bourdot et al. 2010]. However, a fundamental problem is that the geometric primitives (e.g. vertices, edges, triangles) which users interact with are designed for the desktop environment with 2D inputs and might not be suitably controllable in an immersive environment. For example, imagine the user is constructing a 3D architectural model in an immersive environment in the first-person view. It would be difficult for her to reach to the vertices in the corners of the polygon (which are often far away) from her stand point [Salamin et al. 2006] (Figure 1). Some VR systems deploy CSG or B-rep [Ma et al. 2004, Bourdot et al. 2010] primitives in VE to facilitate the modeling process. However, the user controls for these modeling primitives are directly from desktop software; e.g., they only allow 1D or 2D scaling of the selected primitive even though the controllers themselves for the 3D environment provide higher DoFs.

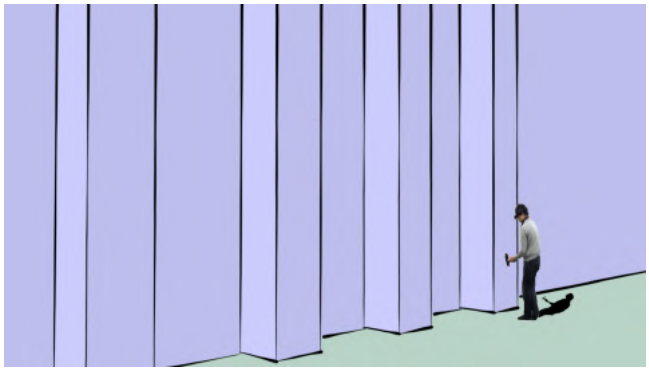

Figure 1: *Illustration of a user using faceton in an immersive environment. The user can easily create walls without worrying about reaching vertices or edges far away.*

In this paper, we present *faceton*, a geometric modeling primitive designed for 3D architectural model using a 6-DoF input device in an immersive virtual environment. A faceton is an oriented point in 3D space and represents an infinite plane passing through it. Edges and vertices are defined as intersections of planes associated with the facetons. The planarity of a face is guaranteed throughout the process, even if the user relocates and rotates a faceton. In the VE, the faceton moves as if it is an oriented disk attached to the front of the 6-DoF controller and users can easily modify its position and orientation through the controller. Figure 1 shows an illustration of a user placing a faceton in an immersive virtual environment. Users can easily put walls in front of themselves without worrying about dragging edges or vertices that could be far away. Figure 2 shows snapshots of an example modeling session in which the user constructed a 3D architectural structure using facetons. In this example, the user built the model shown in Figure 2 with only seven actions, which is significantly fewer than any other modeling method.

The main contribution of the paper is on the vision of combining face primitives with a 6-DOF input device for fast prototyping of architectural buildings in an immersive virtual environment. We describe the interactions for manipulating *faceton* and discuss the pros and cons we observed during a pilot user study. The secondary contribution is an adaptive face-primitives bounding algorithm that drives the *faceton* manipulation at an interactive speed. Interaction speed is essential in VR. To ensure it, the algorithm has several

*e-mail:naoki.sasaki@ui.is.s.u-tokyo.ac.jp
†e-mail:ht.timchen@gmail.com
‡e-mail:d.sakamoto@acm.org
§e-mail:takeo@acm.org

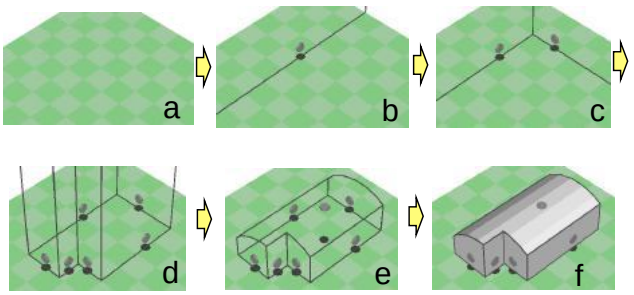

Figure 2: *Modeling with facetons, which are face primitives with three-dimensional position and orientation information defining a planar surface of infinite extent. (a-d) The user designs a model by placing facetons in the virtual environment. (e) The user can also place a primitive representing a cylindrical surface. (f) The system constructs a polygonal mesh model by bounding the surfaces with their intersections.*

limitations and is heuristic, unlike a formal CSG approach; but we believe it forms the basis for future development.

2 Related Work

3D Modeling in a Desktop Environment Various modeling methods have been proposed in the past for 3D CAD such as polygonal meshes, solid modeling, and parametric (feature-based or history-based) representation. However, most of these methods are designed for careful editing of large and accurate 3D models ready for manufacturing and impose too much overhead to permit quick exploration of simple geometric forms. In contrast, our goal is to support very rapid instantiation of rough geometric shapes while sacrificing accuracy and scalability to some extent.

Our work is related to sketching systems in that we aim at rapid instantiation of rough forms. Typical sketching systems [Lipson and Shpitalni 1996, Zeleznik et al. 1996, Schmidt et al. 2009] take 2D lines representing edges in the 3D scene as input and leverage 2D input devices such as pens and touch pads. Whereas, our method takes 3D face primitives as input and takes advantage of 6-DoF input devices. Our work can be seen as an effort to support sketching of architectural forms in an immersive environment.

Push-pull operations introduced by SketchUP [Schell et al. 2000] can be also useful in immersive environments so that the user can directly grab and drag a face. However, the push-pull operation is provided as a short cut of vertex based modeling and needs to be used in combination with other operations. For example, if one wants to put a column at the corner of a box (Figure 2f), the user first has to draw a rectangle on the ground before pulling up the rectangle vertically. This is problematic for immersive environments where is it tedious to switch between different tools (drawing and push-pull) and to reach the ground to draw the rectangle.

3D Modeling in Virtual Environment Many studies have been done on 3D modeling in an immersive environment. Closely related to our architecture building scenario are those aimed at integrating CAD design into the virtual environment: Butterworth et al. [Butterworth et al. 1992] was the first to show the potential of CAD modeling with a head mounted display (HMD) in an immersive environment. Fernando et al. [Fernando et al. 1999] proposed a software framework for constrained VE, whereas Ma et al. [Ma et al. 2004] discussed interaction and model representation in solid modeling in VE. The Environ system [Raposo et al. 2009] enables real-time interaction and navigation of industrial CAD objects in VR.

Researchers have also proposed different interaction paradigms for modeling in VE. Tsang et al. [Tsang et al. 1998] linked virtual hand gestures to basic 3D modeling operations for modeling in a VRML world, whereas Matsumiya et al. [Matsumiya et al. 2000] devised a way to learn from the clay work metaphor and designed a free-form modeling system. Anderson et al. [Anderson et al. 2003] let users design architecture using a kiosk toolbox, and Bourdot et al. [Bourdot et al. 2010] proposed a VR-CAD environment with multimodel immersive interactions including a 6-DoF controller, speech, and gestures. Lau et al. [Lau et al. 2012] presented furniture modeling with tangible primitives in augmented reality environment

Some of these studies allow users to control B-Rep primitives, which have similar behavior to the proposed faceton, in VE [Ma et al. 2004, Bourdot et al. 2010]. Yet the fundamental control in the B-Rep remains the same as its desktop counterpart; that is, users can only control one DoF (e.g. move, scaling in a single direction) or perform basic object add / subtraction movements. On the contrast, faceton is a 3D modeling primitive specifically designed for real-time interaction and 3D modeling in an immersive environment with a 6-DoF control.

Constructive Solid Geometry The faceton representation is inspired by classical studies on CSG [Hagen and Roller 1991] and the B-rep technique [Stroud 2006]. A survey of these two disciplines is out of the scope of this paper; here, we only briefly compare faceton with other primitives in CSG and B-rep. The faceton is related to CSG primitives in that edges and vertices are represented as intersections among primitives. Moreover, the concept of a bounding representation is similar to B-rep. However, the important difference is that the extent of our surface primitive (faceton) is adaptively bounded by other surrounding surfaces, whereas each CSG or B-rep primitive usually has a fixed extent. Some systems allow CSG and B-rep primitives to have infinite extents, but their influence is infinite and is not be bounded by other primitives. We believe such an adaptive bounding feature makes faceton easier to understand and more accessible to users who would like to make 3D architecture designs in VE.

3 Prototype System

We built a prototype VR system for testing the usability of the faceton primitive in VE. The VR system consisted of a head-mounted display (Sony HMZ T1), a Wii remote, and an optical motion capture system with 14 cameras covering an area of 2.5 m (W) x 3 m (D) x 2 m (H) (Figure 3). The 3D positions and orientations of both the HMD and Wii remote were tracked by the mocap system at 100 Hz with a 1 mm error. The system was implemented using Java and running on a 1.6 GHz CPU and 4-GB RAM PC. The unoptimized code ran at an interactive speed for all 3D models described in the paper.

4 User Interaction

In our prototype system, 3D architectural models are constructed by placing facetons in a 3D immersive virtual environment with a 6-DoF controller. The currently selected faceton is visualized as a yellow disk and it is moved as if it is attached to the top of the controller (Figure 4c). After being placed in 3D space, the faceton becomes gray, and a planar surface of infinite extent is created (Figure 4b).

Figure 2 shows snapshots of a complete 3D building session from a God's eye view. The user placed a faceton in the VE (Figure 2b),

Figure 3: *A user wearing an HMD and holding a 6 DoF controller. Both the HMD and controller have additional optical markers attached, and their 3D positions and orientations are tracked with a mocap system.*

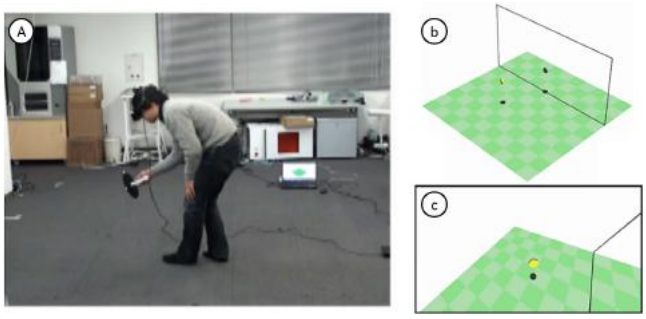

Figure 4: *Snapshot of a modeling session: (a) the user, (b) VE from a God's eye view, and (c) HMD view. The faceton is visualized as a disk floating in 3D space.*

and a planar surface of infinite extent was created. When the user placed the second faceton, another infinite plane appeared (Figure 2c), and the boundaries of both surfaces were updated. Figure 2d shows an intermediate state of the 3D building consisting of six facetons. During the modeling process, the user could grab the existing facetons and arbitrarily modify their position and orientation. Their corresponding surface boundaries were concurrently updated and the planarity of the faces was always preserved.

Note that in our prototype, users only had access to the faceton primitive because we wanted to collect more focused and meaningful feedback. However, since the polygonal mesh model was interactively constructed from the facetons, our system can also support standard mesh manipulation tools, e.g., vertex dragging and edge adjustment.

4.1 Predefined Faceton Group

To facilitate the building process, our system provides several predefined faceton groups. Each group consist of a number of facetons having a certain spatial arrangement. In particular, the dual-faceton group contains two perpendicular facetons, and users can use it to create a two-sided beam into a building (Figure 5b), triple-faceton groups can be used to create three-sided beams (Figure 5c) and corner tables (Figure 5d), whereas quad-faceton groups can be used to create a four-sided pillar.

We also provided predefined cylindrical-faceton groups, wherein a cylindrical surface is approximated by a group of facetons. In Figure 2e, the user has added a curved roof to the building by using the cylindrical-faceton group, while in Figure 5f, the user has added a rounded pillar. Users can also control the scale and radius of these predefined faceton groups via the controller.

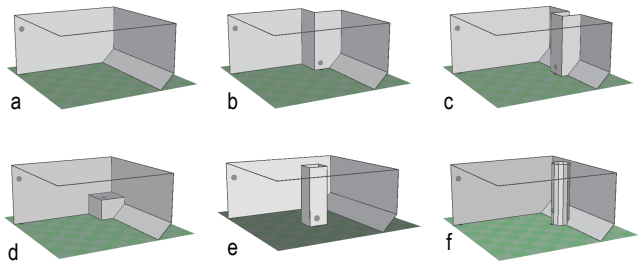

Figure 5: *Adding a feature to the scene by placing a predefined faceton group. (a) Initial scene; (b) two-sided beam made with dual-faceton group; (c) three-sided beam made with triple-faceton group; (d) corner made with triple-faceton group; (e) pillar made with quad-faceton group and (f) round pillar made with cylindrical-faceton group.*

4.2 Customizable Faceton Group

In addition to the predefined faceton groups, the user can also manually group multiple facetons together and manipulate them as a single entity (Figure 6). The customized faceton group is visualized in red, and users interact with the faceton group via a representative faceton, which we called the *kernel* (e.g. the gray faceton on the group in Figure 6a). Using the *kernel* as a handle, users can move, rotate or delete the customized faceton group with the 6-DoF controller.

5 Algorithm

This section discusses how to compute the polygonal mesh model from the given set of facetons. We also discuss how to extend the basic algorithm to support a scene defined as a hierarchy of faceton groups. These aspects turned out to be more difficult than they had first appeared, and the solutions presented here are still exploratory in nature.

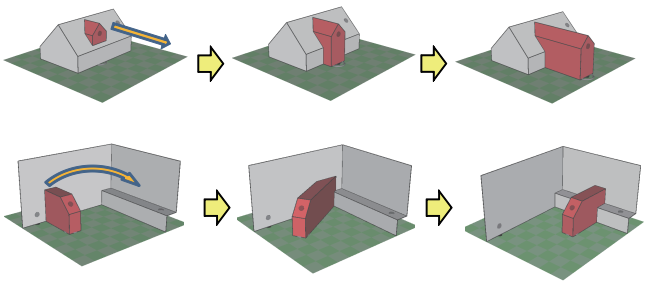

Figure 6: *Modeling operations using faceton groups. The user can relocate (top) and rotate (bottom) a set of facetons together as a group.*

5.1 Bounding Plain Facetons

We investigated several possible designs to convert a set of facetons into a polygonal mesh before settling on the current design. We describe these alternative designs, together with their limitations, and then introduce the current design.

A naïve approach is to divide all of the planes defined by the facetons at their intersections, and then pick the regions that contain the source facetons. Figure 7b illustrates this design in 2D. The definition is clear and the implementation is straightforward. It works perfectly well for convex models. However, this algorithm can produce counterintuitive results for concave models because it causes open boundaries, as shown in Figure 7b (bottom). Open boundaries are not desirable because it is not immediately obvious which facetons define the boundary.

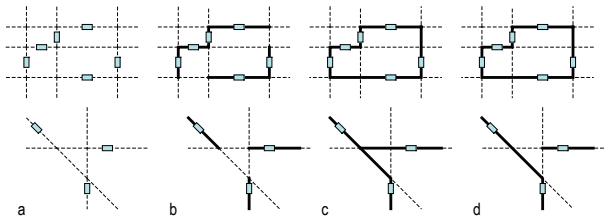

Figure 7: *Possible algorithms for bounding planes, showing two separate examples in different rows. All faces are vertical, and the figures show top-down views. (a) Input facetons. (b) naïve method. (c) Expansion until no open boundary remains. (d) Our method.*

A possible solution is to collect surrounding regions until no open boundary remains (Figure 7c). This method can eliminate open boundaries in the scene. A problem is that the result depends on the order in which the facetons are processed. Figure 7c (bottom) shows the result of expanding the top-left faceton first, but this may be different if the system expands another faceton first. This is problematic because the order of facetons is not readily discernible to users and specifying the order can be tedious. It might be possible to visualize this order information, but the visualization would clutter the screen and the relationship between the order and the final geometry may be difficult to understand.

Another possibility is to explicitly represent connectivity among facetons. This is similar to the standard polygonal mesh representation, where connectivity among vertices is explicitly defined as edges. However, it is too tedious to manually define connectivities for all faceton pairs, especially because our aim is rapid modeling using a 6 DOF input device. Furthermore, the idea of connectivity is insufficient to uniquely define polygonal mesh computation from a given faceton set. For example, the three models shown in Figure 8 cannot be distinguished by connectivity alone (faceton A, B, and C are all connected in these models).

Based on these considerations, our current design uses the distance between faceton pairs as a key. First, we enumerate all faceton pairs in the scene. Then, we put all of the facetons in a priority queue based on the Euclidian distances between the facetons in 3D space. In the main loop, we retrieve faceton pairs with the minimum distance one-by-one and bound them at their intersection. Figure 9 illustrates this process. This strategy can handle both convex and concave shapes appropriately.

Most importantly, the result is fully defined by the visible geometric information (spatial position of facetons) and is not dependent on hidden information such as an order or a dependency graph. If the result is not satisfactory, the user can search for the desired result

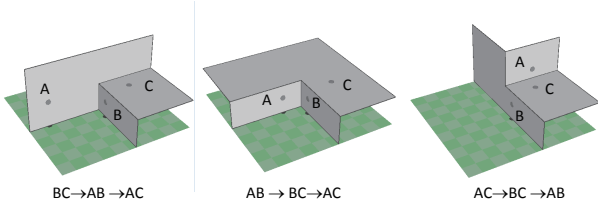

Figure 8: *The user controls the bounding process by adjusting the spatial relationships among facetons. The three facetons define an identical set of infinite planes, but the bounding results are different because the distances among facetons are different.*

just by changing the positions of the facetons. Figure 8 shows an example where the user obtains different results by changing the relative position of the facetons. The infinite planes defined by the facetons are the same, but the final results are different because the relative positions of the facetons are different. This may sound difficult to control, but users can use faceton groups when they want to have more explicit control.

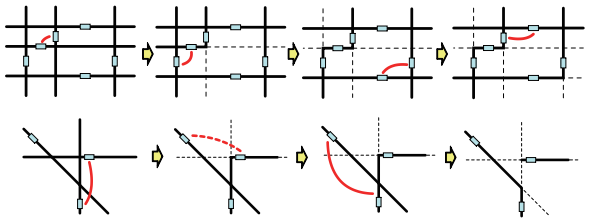

Figure 9: *Our current algorithm for bounding planes. The system enumerates all faceton pairs and bounds the pairs one-by-one starting from the closest pair. Nothing occurs when the pair does not have any intersection.*

As shown in Figure 9 (bottom), this may generate an open boundary at an edge. However, despite that this situation is ambiguous, we believe that the result is an acceptable compromise. The user can easily obtain other results by changing the positions of the facetons. Another shortcoming of our method is that a plane is always bounded at the first intersection, so it does not stretch beyond the intersection even when the user wants a T-shaped junction. However, one can generate a T-junction by placing another faceton on the other side of the intersection.

Our current implementation first bounds the infinite plane defined by each faceton with the ground and a sufficiently large box covering the entire scene. This makes it possible to represent a bounded plane associated with a faceton as a simple polygonal face at any time during subsequent computations.

5.2 Bounding Faceton Groups

A group can contain another group as a member, so the entire scene can be represented as a tree. Each group is associated with a unique polygonal mesh. The system recursively constructs the polygonal mesh in a bottom-up manner from leaf nodes to the root node, and the model associated with the root is presented to the user as the final result. The user can control the final geometry using the group structure. Figure 10 shows some simple examples where the final polygonal mesh differs depending on the group structure.

The process described in the previous subsection is equivalent to computation of a polygonal mesh at the leaf nodes. The system

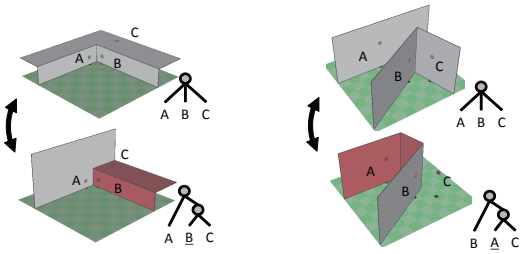

Figure 10: *Different polygonal meshes are obtained because of the difference in the group structure. In each example (left and right), positions of facetons are the same in the top (without groups) and bottom (with groups) figures. The trees show the group structures and the underlined symbol represents the kernel of the group.*

sorts all of the faceton pairs within the group by distance and bounds the planar faces associated with the faceton pairs one-by-one. In the intermediate nodes, the process is generalized to bounding among polygonal meshes defined by the faceton groups. It starts with the sorting of faceton group pairs as in the bounding of plain facetons. The system processes pairs of groups one-by-one in ascending order of distance. The question then becomes how to process a pair of faceton groups, that is, how to bound the two polygonal meshes associated with the pair.

Our current strategy is to segment the two polygonal meshes with their intersections, and collect the segmented polygonal faces by propagation (Figure 11). The propagation starts with the kernels of the faceton groups and stops at an intersection (Figure 11b,c). If the intersection is open (the propagation on the other polygonal mesh has not arrived at the intersection), the propagation continues beyond the intersection until it is bounded by a closed intersection (Figure 11d,e). Whenever a propagation front reaches an open boundary, it pauses and is stored in a priority queue. The queue is sorted on the basis of the geodesic distance from the kernel. The system repeats the propagation by retrieving a propagation front from the priority queue until it becomes empty.

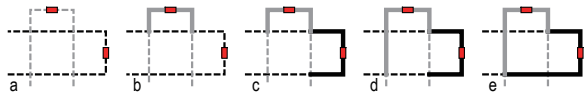

Figure 11: *Bounding of group pairs. Red boxes represent kernels.*

The above strategy generates plausible results in most cases. However, it is not perfect, and it can produce counterintuitive results when polygonal meshes intersect many times (Figure 12). This is not a serious problem in our experience because the users usually set a meaningful feature as a group and such a folded surface rarely appears. Nonetheless, this is one of the problems we want to solve in the future.

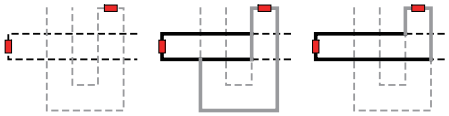

Figure 12: *A problematic result obtained by our current algorithm for bounding a group pair. Left: input groups. Center: result obtained by our current algorithm. Right: a better result, but it is not obvious how to compute this.*

6 Initial Evaluation and Results

An initial pilot user study was conducted to evaluate the usability of faceton in VE. We recruited eight participants who had knowledge about 3D graphics and 3D modeling from the computer science department at University of Tokyo. The user study began with a ten-minute introduction to the faceton primitive and the prototype VR system, followed by a five-minute practice session where we equipped the participant with an HMD and 6-DoF controller and let him or her freely test all provided functions. After the practice session, we asked participants to create an architectural building from scratch using only facetons primitives. Finally, we conducted a brief interview and collected feedback from the participants.

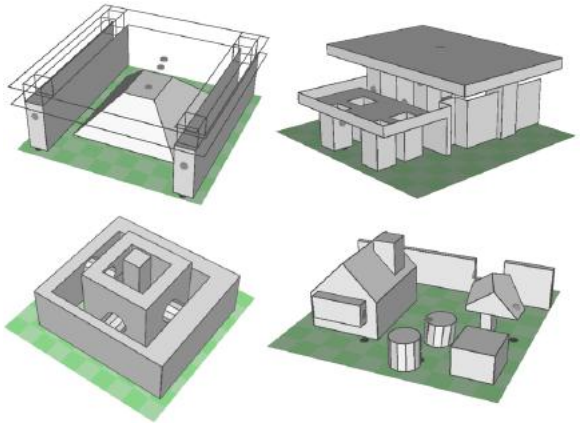

Figure 13: *3D architecture buildings created by the participants.*

Figure 13 illustrates some of the 3D buildings created by the participants. We were very excited to see that participants could build buildings with fairly complex structures by using our prototype system. During the interview, we also learned that most participants enjoyed the construction experience in an immersive environment and agreed that faceton primitive is easy to understand and use in a VE.

However, the participants also raised several usability issues. A number of participants mentioned that sometimes the geometry generated from the facetons was not what they expected and they had to work around it by grouping facetons together. One participant expressed the concern that as the count of facetons and the complexity of the buildings increased, it became difficult to manage or select the desired faceton, especially from the first-person view. One participant also suggested that we should integrate other 3D modeling primitives into the system because he considered it sometimes difficult to add certain details, like bumps and arbitrary curved surfaces, to the building by using facetons. We believed these issues could be addressed with a better faceton management interface or better integration with a more powerful VR 3D modeling system.

7 Conclusion and Future Work

We presented faceton, an interactive geometric modeling primitive designed for building 3D architectural model in an immersive virtual environment. Given a set of facetons, the polygonal mesh model is interactively constructed by a new adaptive bounding algorithm that computes edges and vertices as intersections between the planes defined by facetons. With the 6-DoF controller, users can easily manipule the positions and orientations of the facetons and rapidly construct 3D architectures in VE. Our prototype system

also provides predefined and customizable faceton group functions to facilitate the modeling process.

The algorithm described in section 5 is one possible approach to support the construction of 3D model from a group of facetons. However, it was not easy to devise a perfect algorithm that always generates intuitive results, and the algorithm of our current implementation has several limitations.

First, there is no guarantee that the user can obtain the desired shape by using the presented method. Based on our experience, we believe that our method can represent an arbitrary geometric shape by the user properly placing facetons and combining groups, but a mathematical proof of this is lacking at this point. It is also not clear whether users can find a way to obtain the desired shape by using facetons and groups. These issues will require further investigations.

Second, the polygonal mesh construction can become unstable in certain faceton configurations. For example, when two planes defined by two facetons are nearly coplanar, a slight change in the position and orientation of a faceton can result in a large difference in the final polygonal mesh. This problem is not so apparent when using the snapping mechanism, but it can be an issue when the user wants to have fine-grained control. This issue could be more or less resolved with an assistant snapping function, except when the user desires fine-grained control. In such case, it might be better to introduce a special group that controls nearly coplanar sets of planes as cylindrical surfaces.

Finally, our current method does not distinguish between the inside and outside of a solid. That is, a faceton simply defines a face without defining which side of the face is inside or outside a given structure. An alternative method would be to explicitly define the inside-outside information for each faceton (Figure 14 left). This method might resolve some of the problems discussed so far (e.g., Figure 12). However, we chose our current design after several experimentations because specifying inside-outside information for each faceton is too tedious for quick exploration. Furthermore, in this method, one can easily create a contradictory configuration (e.g., Figure 14 right); in such cases, it is not clear what model to present to the user. Visualization of locally defined inside-outside information might be possible for 2D scenes, but it can be difficult or confusing for 3D scenes. Nonetheless, this might be the right choice for certain application scenarios, and we plan to continue investigating this alternative path in the future.

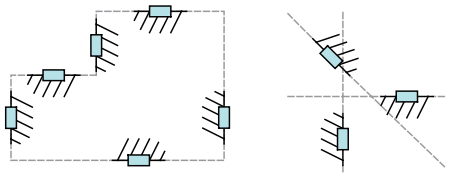

Figure 14: *An alternative method in which the inside-outside information is explicitly defined for each faceton. This method can clearly define a solid for a consistent configuration (left), but the meaning becomes unclear for a contradictory configuration (right).*

Our current prototype system is still a proof-of-concept, but we believe the proposal of this new modeling primitive designed specifically for virtual environments could be beneficial to existing VR tool kits and inspire new research in this direction.

References

ANDERSON, L., ESSER, J., AND INTERRANTE, V. 2003. A virtual environment for conceptual design in architecture. In *Proc. EGVE '03*, 57–63.

BOURDOT, P., CONVARD, T., PICON, F., AMMI, M., TOURAINE, D., AND VÉZIEN, J. M. 2010. Vr-cad integration: Multimodal immersive interaction and advanced haptic paradigms for implicit edition of cad models. *Computer-Aided Design 42*, 5 (May), 445–461.

BRANIT, B., 2007. World builder. http://vimeo.com/3365942.

BUTTERWORTH, J., DAVIDSON, A., HENCH, S., AND OLANO, M. T. 1992. 3dm: a three dimensional modeler using a head-mounted display. In *Proc. I3D '92*, 135–138.

FERNANDO, T., MURRAY, N., TAN, K., AND WIMALARATNE, P. 1999. Software architecture for a constraint-based virtual environment. In *Proc. VRST '99*, 147–154.

GALYEAN, T. A., AND HUGHES, J. F. 1991. Sculpting: an interactive volumetric modeling technique. In *Proc. SIGGRAPH 91*, 267–274.

HAGEN, H., AND ROLLER, D. 1991. *Geometric modeling: methods and applications*. Symbolic computation: Computer graphics–systems and applications. Springer-Verlag.

LAU, M., HIROSE, M., OHGAWARA, A., MITANI, J., AND IGARASHI, T. 2012. Situated modeling: a shape-stamping interface with tangible primitives. In *Proc. TEI '12*, 275–282.

LIPSON, H., AND SHPITALNI, M. 1996. Optimization-based reconstruction of a 3D object from a single freehand line drawing. *Computer-Aided Design 28*, 8, 651 – 663.

MA, W., ZHONG, Y., TSO, S.-K., AND ZHOU, T. 2004. A hierarchically structured and constraint-based data model for intuitive and precise solid modeling in a virtual reality environment. *Computer-Aided Design 36*, 10, 903 – 928.

MATSUMIYA, M., TAKEMURA, H., AND YOKOYA, N. 2000. An immersive modeling system for 3d free-form design using implicit surfaces. In *Proc. VRST '00*, 67–74.

RAPOSO, A., SANTOS, I., SOARES, L., WAGNER, G., CORSEUIL, E., AND GATTASS, M. 2009. Environ: Integrating vr and cad in engineering projects. *Computer Graphics and Applications, IEEE 29*, 6, 91–95.

SALAMIN, P., THALMANN, D., AND VEXO, F. 2006. The benefits of third-person perspective in virtual and augmented reality? In *Proc. VRST '06*, 27–30.

SCHELL, B., ESCH, J. L., AND ULMER, J. E. 2000. System and method for three-dimensional modeling. US Patent 6628279.

SCHKOLNE, S., PRUETT, M., AND SCHRÖDER, P. 2001. Surface drawing: creating organic 3D shapes with the hand and tangible tools. In *Proc. CHI '01*, 261–268.

SCHMIDT, R., KHAN, A., SINGH, K., AND KURTENBACH, G. 2009. Analytic drawing of 3D scaffolds. *ACM Trans. Graph. 28*, 5 (Dec.), 149:1–149:10.

STROUD, I. 2006. *Boundary representation modelling techniques*. Springer-Verlag London Limited.

TSANG, E. K. H., SUN, H., AND GREEN, M. 1998. Virtual world modeler. In *VRST '98*, 179–186.

ZELEZNIK, R. C., HERNDON, K. P., AND HUGHES, J. F. 1996. SKETCH: an interface for sketching 3D scenes. In *Proc. SIGGRAPH 96*, 163–170.