A System for Modelling in Three-Dimensional Discrete Space

Andreas Emmerling, Kristian Hildebrand, Jörg Hoffmann, Przemyslaw Musialski, and Grit Thürmer

Computer Graphics, Visualization, Man-Machine Communication Group
Faculty of Media, Bauhaus-University Weimar, 99421 Weimar, Germany
{andreas.emmerling,kristian.hildebrand,joerg.hoffmann,
przemyslaw.musialski,grit.thuermer}@medien.uni-weimar.de

Abstract. A system for modelling in three-dimensional discrete space is presented. Objects can be modelled combining simple shapes by set operations to obtain large and regular shapes. The system also supports aspects of free-form modelling to generate organic and more complex shapes. Techniques known from image processing are applied to transform and to smooth objects. The basic geometric transformations translation, rotation, scaling, and shearing are provided for discrete objects.

1 Introduction

The growing interest of computer graphics in the three-dimensional discrete space opens up a new application field of volume data: volume-based interactive design and sculpting [10,16]. This requires a new modelling approach based on the discrete space which deals with the generation and manipulation of synthetic objects. If a solid object is represented in continuous space by its boundary surfaces, e.g. by a polygon mesh, the manipulation of local geometric properties may effect the entire surface representation. In contrast, such local manipulations of objects can be easily performed if the objects are represented in discrete space. Moreover, objects modelled in discrete space can be directly merged with measured data, e.g. as obtained from computed tomography. This is frequently necessary in applications of virtual reality in medicine [17].

A number of systems have been already developed for modelling in discrete space, which are mainly concerned with special cases either in the way of modelling [2,4] or in the representation of the discrete space [13,5,9]. Rastering geometric descriptions of continuous objects is one approach. For example, object boundaries are modelled using NURBS [19]. The continuous representation of the boundaries is voxelized to obtain a set of voxels as discrete representation. Modelling with conventional surface based modellers and representing the resulting objects in discrete space tries to make advantage of both kinds of representation [6,19,11].

Less hybrid systems have been developed using *Constructive Solid Geometry* (CSG) [15,2,3], whereas CSG is not always related to classical solid geometry.

I. Nyström et al. (Eds.): DGCI 2003, LNCS 2886, pp. 534–543, 2003.

Instead, simple elementary objects are used to modify a model in the context of free-form modelling by adding voxels to or subtracting voxels from an object by set operations. This modelling technique is often called *volume sculpting*. Free-form modelling is well-suited to model objects which cannot be easily described by basic geometric shapes.

Another enhancement of the CSG approach is the idea of *sweeping* objects. A volume can be created by sweeping a two- or three-dimensional template along three-dimensional trajectories [1,18]. Sweeping gives good results if accuracy is significant since the movement is done along a predefined continuous curve.

There exists another extension of CSG for volume graphics: *Constructive Volume Geometry* (CVG) [8,7] in which the objects are represented by their scalar fields. In case more than one object occupies the same voxel their scalar values, e.g. colours, are adjusted in this voxel.

We have been developing a system for modelling in three-dimensional discrete space to experiment with different modelling approaches. In the future, we want to have a modelling system which relies on the advantages of representing objects by volumetric data combined with a functionality and an easy handling known from modelling in continuous space. The system works consistently in discrete space. We never make use of an immediate continuous representation of the objects as other systems do, e.g. [19]. The first results of our investigation are presented in this paper.

The paper is organized as follows: Section 2 outlines the fundamentals of the modelling system and states our basic assumptions. Section 3 deals with the properties of objects in the system. Section 4 is concerned with the functionality provided by the system to generate and manipulate objects. Afterwards, the implementation and the interface are briefly described in Sect. 5. Finally, Sect. 6 summarizes the paper.

2 Modelling Approach

The n-dimensional discrete space $\mathbb{Z}^n$ is constituted by the n-dimensional array of points with integer coordinates in the Cartesian coordinate system. An object in discrete space is a subset of $\mathbb{Z}^n$. There exists another approach to define objects in discrete space based on the assumption that the discrete space is a tessellation of the continuous space: a point in $\mathbb{Z}^n$ is assumed to represent an n-dimensional unit cube. In $\mathbb{Z}^3$, such a unit cube is called *voxel*. If each unit cube has homogeneous properties, the two representations are basically exchangeable in volume modelling.

We are interested in modelling of solid objects in discrete space, i.e. an object in discrete space has homogeneous properties such that each point of the object has the same properties, e.g. colour. As stated above, an object in discrete space is assumed as a set of points of $\mathbb{Z}^n$. Especially in three-dimensional discrete space, an object is a set of voxels. A scene is a set of objects placed in the discrete space, whereby each point of the space is either empty if belongs to none object, or it is full, then it belongs to exactly one object.

Fig. 1. Compound object.

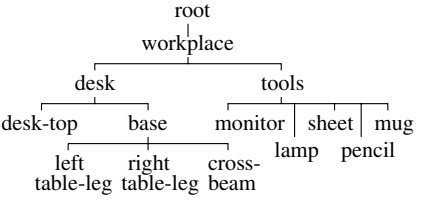

Fig. 2. Object hierarchy.

The system supports two basic modelling approaches: on the one hand, there are CSG tools to generate and manipulate large regular shapes by *set operations* with *basic geometric shapes*. Such shapes require also basic *geometric transformations*, e.g. translation, rotation, scaling, and shearing. On the other hand, there are tools for *free-form modelling* to obtain organic and more complex shapes. These approaches are described in detail in Sect. 4.

Modelling complex objects or scenes containing a rather large number of objects requires the possibility to compound, decompound and recompound objects. For this reason, our system supports an *object hierarchy* which can be dynamically changed by the user.

Morphological operations are provided to smooth small details of objects and to identify the hull of objects. How these operations are applied for modelling is described in the Sect. 4.4.

3 Object Properties

In our modelling system we consider a finite subset V of $\mathbb{Z}^3$ which we want to call *volume buffer* subsequently. More precisely, V is a regular three-dimensional array of points. As stated in the previous section, an object O in our modelling system is a subset of $\mathbb{Z}^3$, which is located in V such that $O \subset V$.

We need some properties of O for the object management and for an efficient processing of O. The set O has a certain *size* $|O|$ which is expressed by the number of points of O. There is no requirement on the connectivity of O. However, connectivity of an object is kept by the transformations. For example, if an object is a connected set and has no holes before a rotation than the object must have the same properties after a rotation. For processing, each object O is associated with a unique *identifier* which is assigned by the system and is saved in the volume buffer at the position of the voxels which belong to O. Additional meta data like *name*, *colour*, and *density* complete the object representation.

Each object has a certain position in the object hierarchy. An object represented by a leaf in the hierarchy is a *simple object* which cannot be decompound

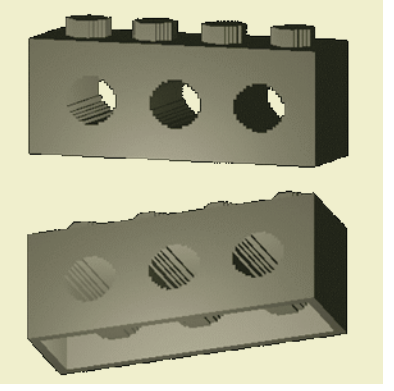

Fig. 3. Views of a Lego block.

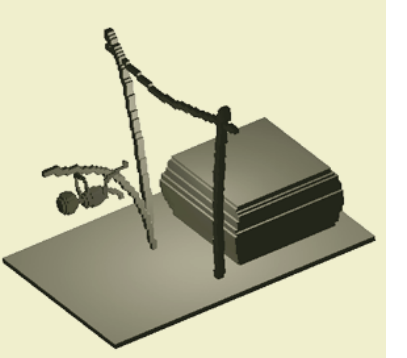

Fig. 4. Olympic games.

further. In contrast, an object is a *compound object* if it is the result of grouping a number of simple or compound objects. Then the object is represented by an inner knot in the hierarchy. The properties of a compound object like size and bounding box depend on the simple objects from which the object is composed. The object hierarchy for the example in Fig. 1 is illustrated in Fig.2. For example, the object *desk* is a compound object and the object *desk-top* is a simple object. The *root-knot* is pre-defined by the system. Any object which is created in the volume buffer is per default a child-knot of the root, i.e. it is a simple object which does not belong to any compound object. The hierarchy can be changed interactively by the user.

4 Generation and Manipulation of Objects

4.1 Generation, Deletion, Set Operations

Simple objects can be generated by rastering basic geometric shapes, e.g. sphere, cylinder, cone or cuboid, and combinations of them. The second way to obtain an object is by setting the voxels of this object one-by-one, which is not very efficient for large objects. A third way is the import of binary data, e.g. from medical imaging, into our system. This data can be manipulated subsequently in the same way like synthetic data. Of course, all voxels of an object can be deleted at once after the object has been selected. Alternatively, each voxel can be deleted separately. The set operations union, difference and average are provided by the system to combine single objects as well as compound objects. An example is presented in Fig. 3 which is the result of combining simple shapes with set operations.

4.2 Templates

Another way of generating an object is to define a three-dimensional template $T \subset \mathbb{Z}^3$, which can be viewed as three-dimensional pencil that is moved in

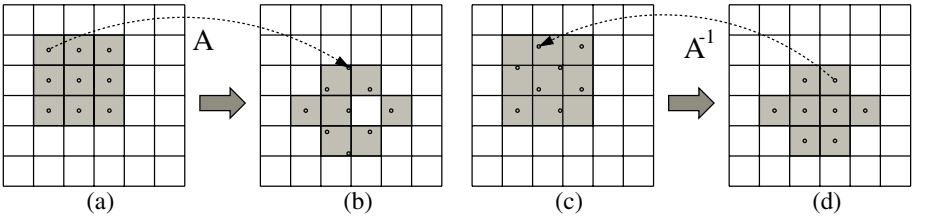

Fig. 5. Rotation by $-\pi/4$: (a) original object O (grey squares), (b) result after rotating each point of O (dots) and O' (grey square) after nearest neighbour rounding, (c) original object O (grey squares) and relevant points of the output image after inverse transformation (dots), and (d) O' (grey squares).

space. The voxels which are hit by moving T belong to the object. A template T can also be used like an eraser. Then the voxels which are hit by moving T are set to empty. Any simple object can be used as template. In this way the user is able to define his own tools by connecting dynamically an object with one of the operations *fill* or *erase*. This is a very natural way of modelling and is well suited for free-form modelling. Figure 4 shows an example for free-form modelling. The poles and the slate in this Figure are modelled by shifting a template.

4.3 Geometric Transformations

Geometric transformations are important for placing and manipulating objects. At the current state, our system provides the affine transformations translation, scaling (and with it also reflection), rotation, and shearing. Applying the corresponding transformation matrix A to each point of an object O works well for the translation. The other transformations cannot be done in this way since holes may appear in the transformed object. This problem is illustrated in Fig. 5 by an example in two-dimensional space: the object O is rotated by $-\pi/4$ around the center of the lower left point of O. Denote the result with O'. In Fig. 5(a) the points of original object O (grey squares) and with their actual representation in $\mathbb{Z}^3$ (little dots) are shown. Figure 5(b) illustrates the result after rotating each point of O separately. The grey squares in (b) show O' after nearest neighbour rounding. There arises a hole in O' which does not exist in the original object.

To solve the problem of holes, we use a common approach of image processing which has been adapted to volumetric image processing [12]. The basic idea is to successively fill in new values at each position of the output image. For this, we have to reverse the transformation by inverting the transformation matrix and apply this matrix A^{-1} to each point of the output image and round the result to their nearest neighbours of $\mathbb{Z}^3$. This is illustrated in Fig. 5(c) and (d) for our example. For simplification only the points of the output image which hit O are illustrated by dots. Figure 5(d) shows O' (grey squares). Note that in O' arises no hole. We have adapted this approach for our purpose keeping in mind that we want to compute these transformations for separate objects only and not for the entire volume buffer.

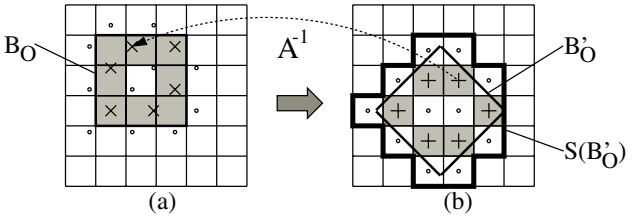

Fig. 6. Rotation by $-\pi/4$: (a) original object O (grey squares) and the points of $S(B'_O)$ after the inverse transformation (dots and crosses), (b) points of $S(B'_O)$ (dots and crosses) and O' (grey squares).

To determine O', one could apply A^{-1} to each point of V. In general, this would be very inefficient. Therefore, the remaining problem is to identify the subset of V, in which the transformed object O' is located. Assume $O \subset V$ and $O' \subset V$. In Sect. 3, it was stated that O is associated with its bounding box B_O which fully includes O. In fact, a bounding box can be viewed as a continuous cuboid. This is illustrated in Fig. 6 for an example in two-dimensional space: again, the object O is rotated by $-\pi/4$ around the center of the lower left point of O and the result is denoted with O'. In Fig. 6(a) the points of O (grey squares) and its bounding box (thicker lines) are illustrated. Apparently, O' will be also enclosed by the cuboid which represents its bounding box after this cuboid is transformed. Let us denote this transformed cuboid with B'_O. We only have to transform the corner points of B_O with A to obtain B'_O. Then we determine the points of V which belong to the supercover of B'_O including the interior points. These points of the example in Fig. 6(b) are surrounded by a thick line. Denote this set of points $S(B'_O)$ and assume $S(B'_O) \subset V$. Finally, we calculate the inverse transformation of each point $q \subset S(B'_O)$. If the nearest neighbour of qA^{-1} is in O then $q \subset O'$. In Fig. 6, the points of $S(B'_O)$ which belong to O' are marked with crosses.

This basic approach for transforming an object in $\mathbb{Z}^3$ can be applied whenever the inverse transformation is known. One should keep in mind, that this is not always the most efficient way, like for the translation. The examples shown in Fig. 7 and Fig. 8 are modelled with simple geometric shapes which were transformed by rotation, shearing, scaling, and translation.

4.4 Morphological Operations

Our system provides a *smoothing* function to reduce tiny details of objects. We apply *morphological filtering* well-known from image processing for this purpose [14]. A morphological transformation is given by the relation of the set of points in question with another (smaller) set of points called *structuring element*. Morphological filtering in digital image processing is done by the two elementary functions: *erosion*, *dilation*, and combinations of them. Erosion shrinks objects by smoothing away the boundaries of an object. Dilation expands objects, fills small holes and connects disjoint parts of an object. Combinations of these functions are used to smooth objects. These combinations are *opening* and *closing*.

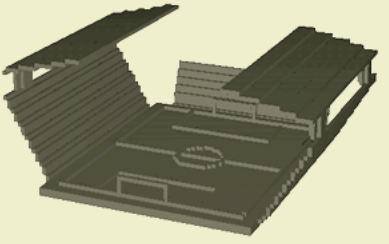

Fig. 7. Stadium.

Fig. 8. Starship.

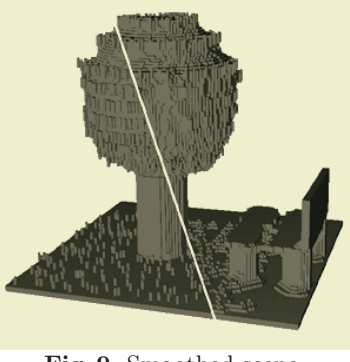

Fig. 9. Smoothed scene.

Fig. 10. Hollowed cylinder.

Opening is defined as erosion followed by dilation and closing as dilation followed by erosion. We applied a discrete sphere as structuring element, i.e. a voxelized sphere with a user-defined diameter. However, other structuring elements, e.g. cubes, could be used as well. Figure 9 shows an example for the result of a morphological closing. The lower left side of the image shows the scene before smoothing and the upper right side shows the result after the closing.

Furthermore, we use the result of the erosion to *hollow out* objects, i.e. to delete the interior of an object. As said above, erosion removes the boundary of an object, i.e. some kind of shell is subtracted from the original object O. The thickness of the shell depends on the structuring element which is applied for the erosion. To hollow out O, the result of the erosion is subtracted from O such that the shell of O is kept. The example of a hollowed cylinder with bases removed is shown in Fig. 10. The left side of the image shows the shell after hollowing the entire cylinder and the right side shows the result after hollowing the shell with a smaller structuring element.

The morphological operations as described above turned out to be a valuable tool for modelling. The smoothing is particularly important for free-from modelling and hollowing out objects is frequently useful for modelling regular shapes.

5 Implementation and Interface

For the implementation of our modelling system, we decided to rely on a common PC with the ordinary input devices mouse and keyboard. We give a brief outline of the implementation below. Basically, the system is subdivided into four parts:

- The *volume buffer and the data management* are responsible for the entire data handling including the organization of the data flow and the memory management for a session.
- The *interface* is implemented using QT 3.0. It enables to model interactively via menu and icon-based control. Sessions and scenes can be stored on external storage units. The main parts of the interface are illustrated in Fig. 11.

  We tried to compensate the disadvantages of the input devices to interact with a three-dimensional scene by an interface which enables a straightforward navigation in and manipulation of the volume buffer. The user is able to move three slices through the volume buffer that are parallel to the three coordinate planes. Objects can be interactively selected and placed in this slices. The slice-based interaction is a first and easy way to model in discrete space. However, it is not always sufficient. Therefore a numerical manipulation of objects is also supported: a command line interface is provided and parameters, e.g. for the transformations, can be set numerically.

  A real-time OpenGL-based visualization unit gives the user visual feedback. Like in the examples shown throughout the paper, the objects are visualized by their cuberille representation. We have preferred this representation for the modelling process since the results can be well judged on the voxel level.
- The *interaction pool* keeps track of all provided interactions between the user and the volume buffer, and does their execution. The result of an interaction is mapped directly into the volume buffer.
- The *object pool* manages the object hierarchy and the meta-data associated with each object and organizes the data flow between the volume buffer and the interactions.

6 Summary

We have presented a system for modelling in three-dimensional discrete space. Our system is intended as experimental environment for volume modelling. It enables different modelling approaches and works consistently in discrete space. The combination of simple shapes is suitable to generate large and regular shapes. Free-form modelling is supported by defining templates and moving them interactively in the volume buffer. The management of complex objects is facilitated by an object hierarchy. The geometric transformations translation, rotation, scaling, and shearing are provided for objects in discrete space. Morphological operations to smooth and hollow out objects turned out to be a powerful tool for any modelling approach. At the current state, the functionality provided

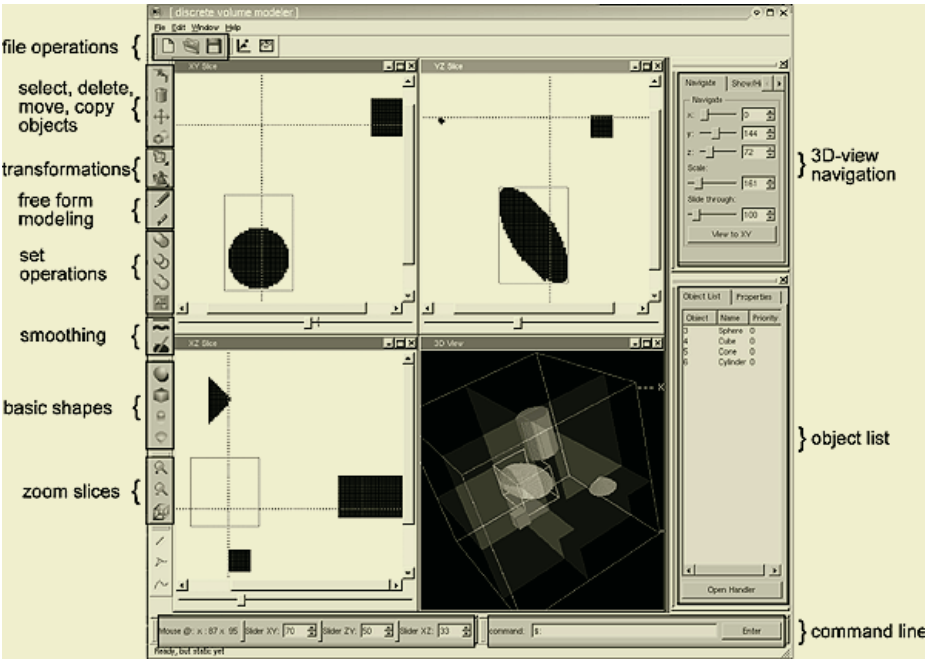

Fig. 11. Interface.

by the system already supports the modelling of a wide variety of objects. This is illustrated by the examples shown throughout the paper.

The development of our modelling system is in progress. In a next step we are concerned with deformations. At the current state, the interface is not comfortable for an unexperienced user. It will also be a matter of future work to investigate in the development of a more intuitive interface with an improved visualization unit.

Acknowledgement

We would like to thank all people who have contributed to the development and the implementation of the modelling system. In particular, thanks are given to Sebastian Derkau and Marcel Schlönvoigt. Thanks also go to Christoph Lincke and Marko Meister for helpful discussions and comments on the paper.

References

1. Ayasse, J., and Müller, H. Interactive manipulation of voxel volumes with free-formed voxel tools. In *Vision, Modeling, and Visualization 2001*, T. Ertl, B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, Eds. IOS Press - infix, 2001, pp. 359–366.

2. Bærentzen, A. Octree–based volume sculpting. In *LBHT Proceedings of IEEE Visualization '98* (October 1998), C. M. Wittenbrink and A. Varshney, Eds.
3. Bærentzen, A., and Christensen, N. J. A technique for volumetric CSG based on morphology. In *Volume Graphics 2001*, K. Mueller and A. Kaufman, Eds. Springer–Verlag, 2001, pp. 117–130.
4. Bærentzen, J. A., and Christensen, N. J. Volume sculpting using the level-set method. *Shape Modeling International, 2002. Proceedings* (2002), 175–182.
5. Bönning, R., and Müller, H. Interactive sculpturing and visualization of unbounded voxel volumes. In *Proceedings 7th ACM Symposium on Solid Modeling and Applications* (2002), pp. 212–219.
6. Chen, H., and Fang, S. A volumetric approach to interactive CSG modeling and rendering. In *Proceedings 5th ACM Symposium on Solid Modeling and Applications* (1999), pp. 318–319.
7. Chen, M., and Tucker, J. V. Constructive volume geometry. *Computer Graphics Forum 19*, 4 (2000), 281–293.
8. Chen, M., Tucker, V., and Leu, A. Constructive representations of volumetric environments. In *Volume Graphics*, M. Chen, A. E. Kaufman, and R. Yagel, Eds. Springer–Verlag, 2000, pp. 97–117.
9. Chen, M., Winter, A. S., Rodgman, D., and Treavett, S. M. F. Enriching volume modelling with scalar fields. In *Data Visualization: The State of The Art*, F. Post, G.-P. Bonneau, and G.Nielso, Eds. Kluwer Academic Press, 2002.
10. Kaufman, A., Yagel, R., and Cohen, D. Modeling in volume graphics. In *Modeling in Computer Graphics - Methods and Applications*, B. Falcidieno and T. L. Kunii, Eds. Springer–Verlag, 1993, pp. 441–454.
11. Liao, D., and Fang, S. Fast volumetric CSG modeling using standard graphics system. In *Proceedings 7th ACM Symposium on Solid Modeling and Applications* (2002), pp. 204–211.
12. Lohmann, G. *Volumetric Image Analysis.* Wiley–Teubner, 1998.
13. Savchenko, V. V., Pasko, A. A., Sourin, A. I., and Kunii, T. L. Volume modelling: Representations and advanced operations. In *Proc. of Computer Graphics International '98* (1998), IEEE Computer Society Press, pp. 616–625.
14. Sonka, M., Hlavac, V., and Boyle, R. *Image processing, analysis, and machine vision.* PWS Publishing, 1999.
15. Wang, S., and Kaufman, A. Volume sculpting. In *Symposium on Interactive3D Graphics* (1995), ACM Siggraph, pp. 151–156.
16. Wang, S. W., and Kaufman, A. E. Volume-sampled 3D modeling. *IEEE Computer Graphics and Applications 14*, 5 (1994), 26–32.
17. Westwood, J., Hoffman, H., Mogel, G., Phillips, R., Robb, R., and Stredney, D., Eds. *Medicine Meets Virtual Reality 11.* IOS Press, 2003.
18. Winter, A. S., and Chen, M. Image-swept volumes. *Computer Graphics Forum (Proc. Eurographics'02) 21*, 3 (2002), 441–450.
19. Wu, Z., Seah, H. S., and Lin, F. NURBS volume for modelling complex objects. In *Volume Graphics*, M. Chen, A. E. Kaufman, and R. Yagel, Eds. Springer–Verlag, 2000, pp. 159–167.