IMPLEMENTATION OF 24-ARY GRID REPRESENTATION FOR RECTANGULAR SOLID DISSECTIONS

Tomokazu Arita[1], Satoshi Kishira[2], Tomoe Motohashi[3], Kenshi Nomaki[2]
[1] *Division of Integrated Sciences, J. F. Oberlin University, Machida, Tokyo, Japan*
[2] *Department of Computer Science and System Analysis, Nihon University, Setagaya, Tokyo, Japan*
[3]*Department of Network and Multi-Media Engineering, Kanto-gakuin University, Yokohama, Kanagawa, Japan*

Kimio Sugita[4], Kensei Tsuchida[5], Takeo Yaku[2]
[4] *Department of Mathematics, Tokai University, Hiratsuka, Kanagawa, Japan*
[5] *Department of Information and Computer Sciences, Toyo University, Kawagoe, Saitama, Japan*

Keywords: Solid and heterogeneous modeling, Octgrids, Tetraicosa-grids, Data structures.

Abstract: In this paper, we propose 24-ary grid graphs corresponding to rectangular solid dissections for ruled line preserving operations. We also show a data structure called H9CODE that corresponds to the 24-ary grid graphs. Furthermore, we describe a voxel unification method in the 24-ary grid graphs and show that the 24-ary grid graphs are an effective model to represent solid graphics.

1 INTRODUCTION

Rectangular solid dissections are commonly used in solid graphics. As graph representation models of rectangular solid dissection processing, octrees (Jackins and Tanimoto, 1980) are well known. The octree can be used in geometric modeling and space planning. The octree structure is an extension of the quadtree structure for the representation of two-dimensional images. A multi-level boundary search algorithm is developed to incorporate surface information into the octree representation. This algorithm makes the octree representation useful for graphic displays and object recognition tasks.

We have examined ruled line oriented transformation of rectangular solid dissections such as voxel unification in solid graphics. We note that voxel unification is frequently used in Level of Detail (LOD) related operation. We previously proposed octal grids called octgrids (Motohashi et al., 2002; Motohashi et al., 2002; Arita et al., 2004; Akagi et al., 2005) for rectangular dissections, and hexadecimal grids called hexadeci-grids (Kureha et al., 2007) for multilayer rectangular dissections with respect to ruled line oriented operations. Several transformation algorithms for octgrids are more efficient than for quadtrees (Arita et al. 2004).

In this paper, we introduce 24-ary grid graphs called "tetraicosa-grids" (Kureha et al., 2007; Kishira, Tsuchida et al., 2008; Kishira, Kureha et al 2008) that correspond to rectangular solid dissections. Tetraicosa-grid structure is constructed by extending the octgrid structure. We also describe a voxel unification method that runs in O(1) time, and we show a data structure called H9CODE corresponding to the 24-ary grid graphs.

In section 2, we review octgrids for rectangular dissections as preliminaries. Section 3 contains several definitions of tetraicosa-grids. In section 4, we show the H9CODE data format corresponding to the tetraicosa-grids. In section 5, we explain a voxel unification method using H9CODE, and in section 6 we explain the concept of rendering H9CODE.

2 PRELIMINARIES

2.1 Octgrids for Rectangular Dissections

In this section, we deal with heterogeneous rectangular dissections. We review the definitions

Arita T., Kishira S., Motohashi T., Nomaki K., Sugita K., Tsuchida K. and Yaku T.
IMPLEMENTATION OF 24-ARY GRID REPRESENTATION FOR RECTANGULAR SOLID DISSECTIONS.
DOI: 10.5220/0001799201030106
In *Proceedings of the Fourth International Conference on Computer Graphics Theory and Applications (VISIGRAPP 2009)*, page
ISBN: 978-989-8111-67-8

concerning octgrids (Motohashi et al., 2002, Motohashi et al., 2003; Arita et al., 2004; Akagi et al., 2005) that represent rectangular dissections.

Definition 2.1.1

Let $D = (T, P, g)$ be a *rectangular dissection*, where T is an *(n, m) − table* for some n and m, P is a *partition* over T, and g is a *grid* of T. An *octgrid* $G = (V_D, L, E_D, A_D, \alpha_D)$ for D is a multi-edge undirected grid graph, where V_D is identified by partition P (We denote a node corresponding to a *cell* c in P by vc), $L = \{enw, esw, eew, eww\}$, $E_D \subseteq V_D \times L \times V_D$ is a set of undirected labeled edges of V_D of the form $[vc, l, vd]$, where vc and vd are in V_D, and l is in L. Here, E_D is defined by rules 1-4 below, and $A_D = R^4$ and $\alpha_D : V_D \rightarrow R^4$ are defined as the location of perimeter cell c for vc in V_D by $\alpha_D = (nw(c), sw(c), ew(c), ww(c))$.

Rule 1

If $nw(c) = nw(d)$, that is, c and d have a common north wall, and there is no cell between c and d that has an equal north wall, then $[vc, enw, vd]$ is in E_D. In this case, $[vc, enw, vd]$ is called a *north wall edge.*

Rules 2-4

Labeled edges in other directions are similarly defined. The following figure illustrates a rectangular dissection and its corresponding octgrids (Figure 1).

We note the degree of edges is at most eight in octgrids.

2.2 H3CODE (ARITA AND YAKU, 2006)

H3CODE is a data format that represents octgrids. The cell of H3CODE corresponds to a rectangle in a rectangular dissection (See Figure 2(a)). The whole structure of the H3CODE files is shown in Figure 2(b).

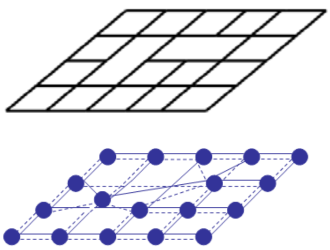
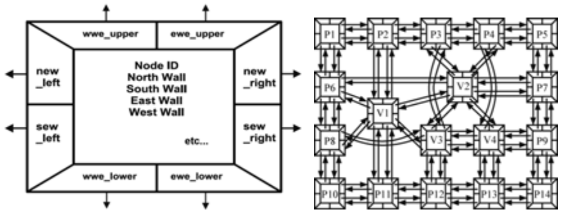


Figure 1: Rectangular dissection (upper) and its corresponding octgrid (lower).



(a) (b)

Figure 2: (a): H3CODE cell, (b): whole structure of H3CODE.

3 24-ARY GRID GRAPHS

Next, we introduce a 24-ary grid graph representation for rectangular solid graphics. Let $D = \{S_1, S_2, ..., S_N\}$ be a rectangular solid dissection, where each S_i is a rectangular solid in D. A tetraicosa-grid for D is an undirected labeled multi-edge grid graph $G_D = (V_D, L, E_D, A)$, defined as follows:

(1) $V_D = \{v_s \mid s$ is in D; v_s corresponds to $s\}$ is a set of *nodes*,
(2) $L = \{$*EquivalentUpwardNorthEastCornerPole, EquivalentDownwardNorthEastCornerPole, ..., EquivalentBackwardFloorWestBeam*$\}$ ($|L| = 24$) is the set of *edge labels*,
(3) E is a set of undirected labeled edges defined as follows; if s and t are the nearest solids in D such that s and t have an upper north beam in common, then [s, *EquivalentForwardCeilingNorthBeam* (Figure 3), t] is in E_D. Edges for other beams and corner poles are similarly defined.

Figure 4 illustrates links around a node in a tetraicosa-grid. Furthermore, Figure 5 shows a rectangular solid dissection (left) and the corresponding tetraicosa-grid (right).

Suppose that D is of a k-width, l-depth, and m-height rectangular solid; let G_D be the tetraicosa-grid for D, and i be the number of inner voxels. We have $2\,|E_D| = 12\times8 + 16\times4(k\text{-}2) + 16\times4(l\text{-}2) + 16\times4(m\text{-}2) + 20\times2(k\text{-}2)(l\text{-}2) + 20\times2(l\text{-}2)(m\text{-}2) + 20\times2(m\text{-}2)(l\text{-}2) + 24\,i$ (Kishira et al., 2008).

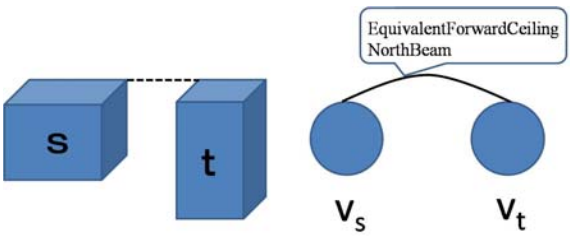


Figure 3: Example of labels for edges.

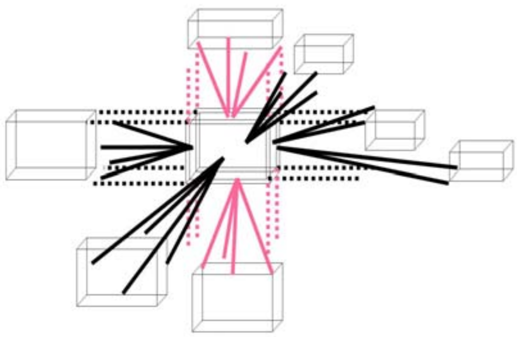


Figure 4: Links around a node in a tetraicosa-grid (Kishira S., Kureha A. et al., 2008).

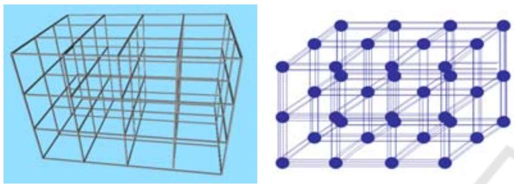


Figure 5: Rectangular solid dissection (left) and its corresponding tetraicosa-grid (right).

4 DATA FORMAT: H9CODE

In this section, we propose a data format H9CODE corresponding to tetraicosa-grids. H9CODE is based on the data format H3CODE (Arita and Yaku, 2006) corresponding to octgrids.

We first show the field numbers with their contents, starting with the 33rd field. We note that fields 1 to 32 in H9CODE are the same as the fields in H3CODE.

33. EquivalentUpwardNorthEastCornerPole
    (See Figure 6 (left))
34. EquivalentDownwardNorthEastCornerPole
33. EquivalentUpwardNorthWestCornerPole
34. EquivalentDownwardNorthWestCornerole
35. EquivalentUpwardSouthEastCornerPole
36. EquivalentDownwardSouthEastCornerPole
37. EquivalentUpwardSouthWestCornerPole
38. EquivalentDownwardSouthWestCornerPole
41. EquivalentForwardCeilingNorthBeam
    (See Figure 6 (right))
42. EquivalentBackwardCeilingNorthBeam
43. EquivalentForwardCeilingSouthBeam
44. EquivalentBackwardCeilingSouthBeam
45. EquivalentForwardFloorNorthBeam
46. EquivalentBackwardFloorNorthBeam
47. EquivalentForwardFloorSouthBeam
48. EquivalentBackwardFloorSouthBeam
49. EquivalentForwardCeilingEastBeam
50. EquivalentForwardCeilingWestBeam
51. EquivalentBackwardCeilingEastBeam
52. EquivalentBackwardCeilingWestBeam
53. EquivalentForwardFloorEastBeam
54. EquivalentForwardFloorWestBeam
55. EquivalentBackwardFloorEastBeam
56. EquivalentBackwardFloorWestBeam

Here, we show examples of H9CODE in Figure 6.

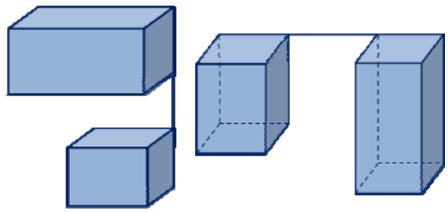


Figure 6: Voxels linked with 33rd field (left) and with 41st field (right).

5 VOXEL UNIFICATION

In this section, we show a voxel unification method with H9CODE. This method unifies properly neighboring voxels and is executed as depicted in Figure 7.

UNIFYVOXEL (See Figure 7)

INPUT

G_D : 24-ary grid graph representation for a rectangular solid dissection D in H9CODE

v_c : a voxel in G_D

v_d : a voxel in G_D; vc and vd have four horizontal beams in common

OUTPUT

G_E : 24-ary grid graph representation for a rectangular solid dissection E in H9CODE, where v_d is unified to v_c.

Method

1. Change links on x-axis in v_d.
2. Change links on y-axis in v_d
3. Change links on z-axis in v_d.
4. Delete v_d.

The UNIFYVOXEL method unifies two neighbour voxels into one voxel by replacing neighbour edges around their voxels based on 24-ary grid graph representations.

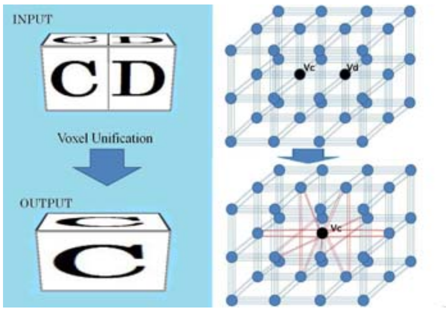


Figure 7: Input and output of voxel unification.

We note that the time complexity of the method is $O(1)$, since the number of links around c and d are bounded by $96 = 48 \times 2$.

6 RENDERING WITH H9CODE

In this section, we present the concepts of rendering programs and an output image created using H9CODE.

First, we show programs to render coordinate values of voxels with VRML and H9CODE.

Program

- ☐ A conversion program of coordinate values of voxels to H9CODE
  - ■ Program Name : cv2h9
  - ■ Input: coordinate values
  - ■ Output: H9CODE
  - ■ Programming language : C

- ☐ A conversion program of H9CODE to VRML
  - ■ Program name : h92vrml
  - ■ Input: H9CODE
  - ■ Output: VRML
  - ■ Programming language : C

An output image of a sphere using the above programs and rendered with H9CODE is shown in Figure 8.

7 CONCLUSIONS

We introduced the basic concepts of a graph representation method called “tetraicosa-grid” for volume graphics. We introduced a data format H9CODE for 24-ary grid graphs. We also showed a voxel unification method using H9CODE. This method unifies two voxels into one voxel and runs in constant time.

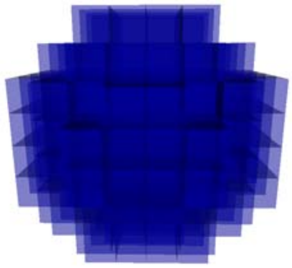


Figure 8: Output image of sphere using above programs with H9CODE.

We are now implementing rendering systems for the 24-ary grid graphs and plan to construct the unification system for the 24-ary grid graphs.

REFERENCES

Arita, T., Tsuchida, K., Motohashi, T., and Yaku, T. (2004). An Octet Degree Graph Representation for the Rectangular Dissections. *Joint Conference on Applied Mathematics*, pages 131-136.

Arita, T. and Yaku, T., (2006). H3-Code 2.3 Reference Manual. http://www.waap.gr.jp/waap-rr/waap-rr-06-001/index.html

Akagi, G., .Motohashi, T., Nomaki, K., and Yaku, T. (2005). Octal Graph Representation for Multi-Resolution 3D Landform Maps and Its Application. In *Proceedings: Applied Mathematics Symposium*, pages 27-32.

Jackins, C. L. and Tanimoto, S. L. (1980). Oct-Trees and Their Use in Representing Three-Dimensional Object. In *Proceedings: Computer Graphics and Image Processing*, Vol.14, No.3, pages 249-270.

Kureha, A., Kishira, S., Motohashi, T., Tsuchida, K., and Yaku, T. (2007). Hexadecimal Grid Graph Representation of Multilayer Rectangular Dissections and Its Applications. 10*th Society for Industrial and Applied Mathematics Conf. in Geometric Design & Computing, Abstract , Texas, USA*, page 32.

Kishira S., Kureha A., Motohashi T., Tsuchida K., and Yaku T. (2008), 24-ary Grid Graph Representation for the Rectangular Solid Dissections, Proceedings of the 2008 IEICE General Conference, page 171.

Kishira, S., Tsuchida, K., Motohashi, T., and Yaku, T. (2008). Tetra-icosa Grid Graph Representation for the Rectangular Solid Dissection. *JSIAM*, pages 65-66 (in Japanese).

Motohashi, T., Tsuchida, K,, and Yaku, T. (2002). Table Processing Based on Attribute Graphs. In *Proceedings: IASTED SEA5*, pages 317-322.

Motohashi, T., Tsuchida, K., and Yaku, T. (2002). Attribute Graphs for Tables and Their Algorithms. In *Proceedings: Foundation of Software Engineering*, pages 183-186.