XVL:
A Compact And Qualified 3D Representation With Lattice Mesh and Surface for the Internet

Akira Wakita [1] Keio University

Makoto Yajima [2] Lattice Technology

Tsuyoshi Harada [2] Lattice Technology

Hiroshi Toriya [2] Lattice Technology

Hiroaki Chiyokura [1] **Keio University**

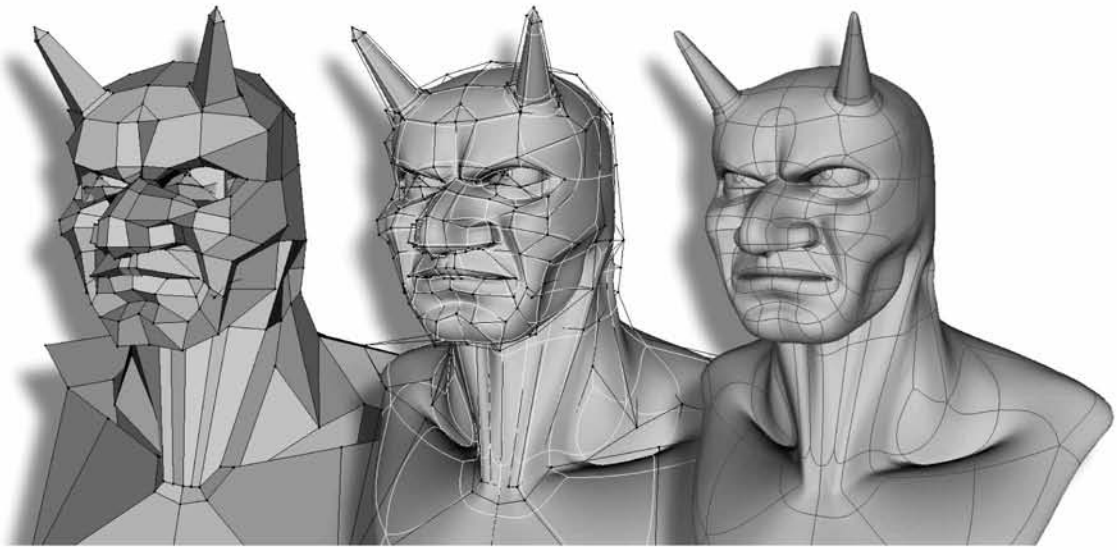

Figure 1: *Lattice structure. Left model is the Lattice Mesh and right model is the corresponding Lattice Surface. Topologies of these models have one to one correspondence.*

Abstract

Computer graphics systems and CAD/CAM systems are widely used and an abundance of 3D-Data in various fields exists. However, based on the VRML technique, it is difficult to send such 3D-Data through the Internet, because of the large data size. Transmission of practical and highly detailed 3D-Data through the Internet becomes a primary requirement. Therefore, a compact and qualified 3D-Data representation method is greatly required. This paper describes XVL (eXtended VRML with Lattice), a new framework for compact 3D-Data representation with high quality surface shape. By utilizing a free-form surface technique, qualified surfaces are transferred with a limited amount of data size and rendered. Free-form surfaces transferred by an efficient data structure are called lattice structure. This data structure contains only vertices, topologies, and attributes, and can be converted to the original surface. Because the lattice structure is regarded as a polygon mesh, it can be easily integrated to a VRML file. These surfaces and lattice have the same topology and are thus interchangeable in the lattice structure. By using weighting attributes, a sophisticated surface shape can be represented. Some practical XVL applications, such as an intuitive surface design system, are also introduced.

[1] Graduate School of Media and Governance, Keio University. 5322 Endo, Fujisawa, Kanagawa, 252-8520, Japan. {wakita, chiyo}@sfc.keio.ac.jp

[2] {yajima, harada, toriya}@lattice.co.jp

CR Categories and Subject Descriptors: I.3.7 [Three-Dimensional Graphics and Realism]: Virtual Reality - ; I.3.6 [Computer Graphics]: Methodology and Techniques - Graphics Data Structures and Data Types; I.3.5 [Computational Geometry and Object Modeling]: Curve, Surface, Solid, and Object Representations - Geometric Algorithms, Languages, and Systems

Additional Keywords: VRML, Geometry Compression, Lattice Mesh

LATTICE TECHNOLOGY

1. INTRODUCTION

VRML offers increased opportunities for the use of 3D-Data through the Internet. Nevertheless, VRML is not as widely accepted as expected and few VRML applications exist. One of the biggest reasons is its data size. When we generate a realistic VRML model from CAD or computer graphics systems, and try to transfer it through the Internet, it takes an enormous amount of time. Widespread use of mid-range CAD and inexpensive CG-systems produced tremendous 3D-Data around the world. If we transfer these 3D-Data rapidly through the Internet, a new use of 3D-Data can be yielded, i.e., 3D digital documents or 3D magazine. Another reason VRML is not widely accepted is that it cannot represent sophisticated surfaces. Primitives and polygon meshes used in VRML cannot represent realistic objects with complicated surfaces. Most 3D CAD/CAM systems employ free-form surfaces like NURBS, which must be converted to polygons when its shape data is transferred as VRML. We need surface representation to popularize the use of VRML in fields like CAD/CAM applications.

A number of methods have been developed to compress the geometry of 3D shapes. Mesh simplification technique is one of them. Hoppe[7] has presented an algorithm for optimizing surface meshes. The main idea is to reduce the number of polygons to minimize energy function. Different from other mesh simplification algorithms, Hoppe's method can reconstruct original shapes from simplified shapes. Lounsbery et al.[8] and Eck et.al.[6] have developed the idea of Multiresolution Analysis. The main idea is that any function can be decomposed to a low resolution function and a series of components. This idea is applied to the meshes of arbitrary topology. Taubin et. al.[9] have presented the Compressed Binary Format of VRML. Polyhedron data is compressed by describing two interlocking trees. Abadjev et. al.[1] have developed MetaStream, a practical framework to deliver 3D-Data with texture data. This approach constructs 3D-Data and additional information progressively. Although many approaches have been proposed, none of them satisfies the requirements of both data compactness and surface shape quality. This is because these methods are based on polygon meshes.

Subdivision surface is another approach to generate a qualified rendering image of surface shape from simple mesh. Doo and Sabin[5] and Catmull and Clark[2] have developed a fundamental theory of this technique. This method can be applied to meshes of arbitrary topology. Our approach, which can generate free-form surfaces from the arbitrary mesh, is an extension of Doo and Sabin's work. DeRose et. al.[4] enhanced Catmull and Clark's method for the practical use of contents creation. Their method can generate a variety of surface shapes by blending multiple surface shapes. Subdivision surfaces are adapted to handle arbitrary topological meshes, which are easy to implement. This method, however, requires successive filtering steps to generate limit surface shape, thus prediction is difficult.

One of our goals is to create compact and qualified 3D-Data suitable for data transmission through the Internet. To achieve this purpose, we propose **XVL(eXtended VRML with Lattice)**, a new framework of 3D-Data using lattice structure. Lattice structure consists of *Lattice Surface* and *Lattice Mesh*. *Lattice Surface* is free-form surface data represented by a Gregory Patch[3]. *Lattice Mesh* is a polygon mesh which has the same topology as *Lattice Surface* and has properties to reconstruct surface shape. In XVL, instead of transferring polygon mesh, *Lattice Mesh*, which consists of vertices, topologies, and weighting properties, is transferred. Because complicated free-form surfaces are generated from simple *Lattice Mesh*, compared to sending polygons, transmission time of the *Lattice Mesh*, is much faster. Based on our invertible rounding algorithm, *Lattice Mesh* can be transformed into *Lattice Surface* and vice versa. Using XVL, we can transfer highly qualified 3D-Data efficiently. To manage lattice structure, we have developed a solid modeling kernel called *Lattice Kernel*. By manipulating simple *Lattice Mesh*, surface shape can be intuitively designed. Therefore, realistic XVL data is easily generated using the aforementioned design systems. Moreover, because XVL is represented as a solid, the data can be directly utilized in the CAD/CAM environment.

In Section 2, the concept of the XVL data format is described and why it is compact. Section 3 presents an algorithm of generating *Lattice Surface* and *Lattice Mesh*. A method to generate various shapes by changing the weighting properties is also described. Section 4 introduces some XVL based applications. In this section, an intuitive surface design system based on the *Lattice Mesh* is described. Practical examples of data exchange between CAD/CAM systems are also presented. Performance issues of the proposed framework are discussed in Section 5. This paper concludes with Section 6, some tasks and future works.

.

2. Data Structure

2.1 Data Structure of XVL

The most important properties of XVL are compact data and surface representation. When a polygon mesh is used to represent a 3D object, to reduce data size, the resulting shape is inadequate. To get high quality data, the data size increases tremendously. A free-form surface is thus used to overcome these problems simultaneously. Free-form surface can represent complicated shapes with few control points. By using free-form surface, the amount of data decreases and more sophisticated shapes can be represented with reality. We have introduced a lattice structure to represent free-form surface efficiently. Using lattice structure makes it possible to reduce the amount of data to represent free-form surface shapes and also to manipulate surface shapes intuitively. Because lattice structure is regarded as a polygon mesh, it can be implemented as an extension of VRML, which we call XVL.

Lattice structure consists of *Lattice Surface* and *Lattice Mesh*. *Lattice Surface* is free-form surface data represented by a Gregory Patch[3]. Gregory Patch has been used to represent surfaces with geometric continuity. Because it can be converted to NURBS, *Lattice Surface* can be used in CAD/CAM systems. Figure 1 shows Lattice structure. Left object is the *Lattice Mesh* and the right object is the corresponding *Lattice Surface*. *Lattice Mesh* is a polygon mesh which has the same topology as *Lattice Surface* and has properties to reconstruct surface shape. As shown in Figure 1, topologies of *Lattice Surface* and *Lattice Mesh* have one to one correspondence and can be interchanged.

Figure 2 shows the method of migrating *Lattice Mesh* to VRML. *Lattice Mesh* can be loaded from any kind of VRML browser as

polygon data. In cases where the browser supports XVL, the polygon will be rounded, thus resulting in free-form surface shape.

```
#VRML V2.0 utf8

PROTO XVL_EDGE[
      field SFFloat round_val 0
      field SFVec3f round_str 0 0 0
      field SFVec3f round_end 0 0 0
      field MFInt32 is_round  [1, 1, 1]
      ]
{
      Text{
         string["weight of edge rounding"]
      }
}

PROTO XVL_STATUS[
      field SFString status "XVL_LATTICE"
      ]
{
      Text{
         string["status of shape"]
      }
}

Group{
      children[

         ## Shape information of Lattice
         Group{
            children[
               Shape{
                  geometry IndexedFaceSet{
                  ############################
                  ## coordinate information          ##
                  ## of Lattice Mesh                 ##
                  ############################
                  }
               }
            ]
         },

         ## Property information of Lattice
         Switch{
            choice[
               XVL_STATUS{
                  status "XVL_GREGORY"
               }
               XVL_EDGE{
                  round_val 0.5
                  round_str 0 1 1
                  round_end 0.2 0.3 1
                  is_round  [1 1 1]
               }
               IndexedLineSet{
                  coordIndex[ 24 103 ]
               }
            ]
            whichChoice -1
         }
      ]
}
```

Figure 2: *Method of XVL. Information of Lattice Mesh is described in Group node. Information of properties to reconstruct Lattice Surface is described in Switch node. In the Switch node, the whichChoice field is set to –1 enabling non-XVL browsers to ignore this field.*

As shown in Figure 2, an XVL file consists of two parts, Group node and Switch node, and combined to one Group node. The Group node contains the information of *Lattice Mesh*, which is equivalent to polygon mesh. The Switch node contains properties necessary to reconstruct *Lattice Surface* from the polygon mesh. In the Switch node, the *whichChoice* field is set to –1 enabling non-XVL browsers to ignore this field. Here, *Lattice Mesh* results in polygon shape on the browser. Otherwise, the browser obtains attribute data from the choice field in order to round the polygon. Also, any type of attribute data which are not supported by VRML can be defined in this field (see PROTO XVL_EDGE in Fig. 2).

The simplest property of this example is version and status information. The status information has two values, XVL_LATTICE and XVL_GREGORY. In the former case, we display the shape with only *Lattice Mesh*. In the latter case, we display the free-form surface shape of *Lattice Surface*. The next property, which can be set to vertices and edges, is weight information of the surface shape. IndexedLineSet node is used to determine to which edges or vertices weight information is applied. The same coordinate of IndexedFaceSet can be used. We describe only the indices in IndexedLineSet. Weight information is described as a scalar value and some vectors. If the property is edge information, we describe one SFFloat field, two SfVec3f fields and one MFInt32 field. The first SFFloat field is weighting value. The next two fields are rounding vector of the start point and rounding vector of the end point. The last field describes the flags, whether to round an edge, start point, or the end point. For example, putting a vertex in this field requires insertion of a weighting value and rounding flags. In the parsing stage of XVL, first we construct the shape of *Lattice Mesh* and then check its properties. If the status information is XVL_GREGORY, *Lattice Surface* is reconstructed from *Lattice Mesh* and weight properties of vertices and edges described here are reflected to the shape of the surface.

2.2 Data Size

VRML makes it possible to transfer 3D-Data through the Internet. Practical VRML data generated from a CAD/CAM system, however, contains large amounts of polygon data tessellated from free-form surfaces. Thus, its data size becomes more than 10MB. Because XVL can represent free-form surfaces, the amount of data required to transfer the same kind of surface shape, becomes much smaller. Figure 3 shows a comparison of the amount of data required by XVL versus VRML. The raptor model is used for comparison in Figure 4. VRML consists of text data, and generally supports binary data compressed by gzip. This also applies to XVL. In Fig. 4, the division number for the polygon is set to 8 for comparison. The result shows XVL can represent the better quality shape with less than 10% the amount of data of VRML. As a result, XVL shows the potential of expanding the capability of 3D-Data use through the Internet. Moreover, after transmitting the XVL file, the division number for the tessellation can be freely edited to obtain required rendering quality. This flexibility is another merit of using XVL.

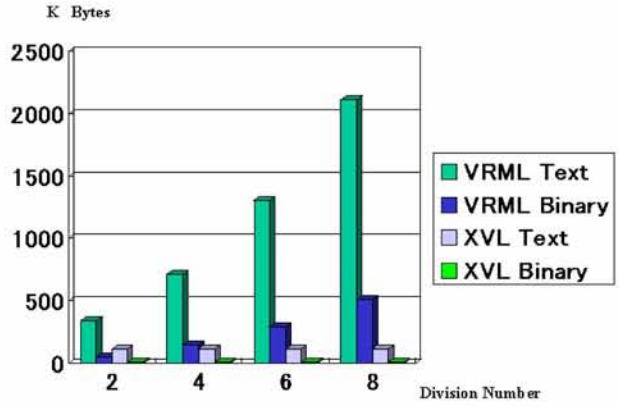

Figure 3: *Comparison of data size. The division number for the polygon is set to 8 for comparison. The result shows the amount of data of XVL is less than 10% of VRML.*

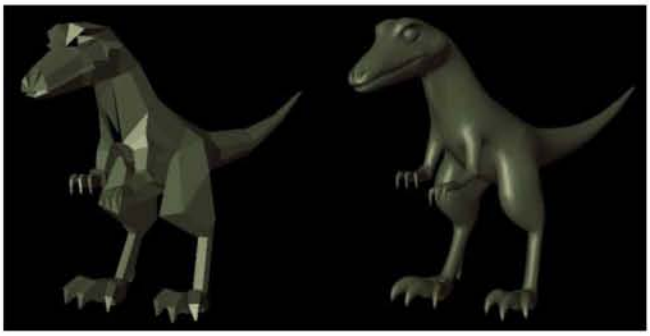

Figure 4: *Raptor model used for comparison. Left model is Lattice Mesh.*

3. Conversion between Lattice Surface and Lattice Mesh

3.1 Overview of Algorithm

Lattice Surface is generated according to the following steps.

Step 1

Calculate the vertex coordinates of *Lattice Surface* corresponding to the vertex coordinates of *Lattice Mesh*. Vertex coordinates can be obtained by the linear transformation of coordinates of *Lattice Mesh*.

Step 2

Calculate the curves of *Lattice Surface* corresponding to the lines of *Lattice Mesh*. New curves are represented as cubic Bézier curves. Control points are also obtained by the linear transformation of coordinates of *Lattice Mesh*.

Step 3

Interpolate Gregory Patches to the region surrounded with cubic Bézier curves. Topology of the *Lattice Surface* is the same as *Lattice Mesh*.

Control points of *Lattice Surface* are calculated as a linear transformation of vertex coordinates of *Lattice Mesh* as follows:

$$P_0 = \sum K_i V_i$$

where P_0 is an obtained coordinate, V_i is a vertex coordinate of *Lattice Mesh*, and K_i is a real number.

Linear transformation generally has its inverse transformation, as long as it is not singular. Therefore, each vertex coordinate of *Lattice Mesh* V_0 can be calculated by the following function:

$$V_0 = \sum I_j P_j$$

Here I_j is also a real number. These two functions indicate that *Lattice Mesh* can be reconstructed from *Lattice Surface*. To generate a polygon model represented by *Lattice Mesh* from a free-form surface model represented by *Lattice Surface*, generated curves are replaced by straight lines. We call these two operations rounding and inverse rounding operations.

3.2 Generation of Lattice Surface

The basic idea of rounding is the same as the subdivision surface of Doo and Sabin[5]. *Lattice Mesh* and *Lattice Surface* correspond to base mesh and limit surface, respectively, of the subdivision surface. However, while the subdivision surface method requires a number of filtering processes to generate smoothly rounded shapes, our rounding operation generates a smooth surface model directly. The following shows the precise steps of rounding.

Step 1

Step1 is to find the vertex coordinates not changed in subdivision filtering.

Subdivide the *Lattice Mesh* M_0 once with Doo and Sabin's subdivision rule and get the subdivided mesh M_1. Calculate the face points(average of each vertex coordinates of the face) P_0 of M_1. These points become the vertices on *Lattice Surface* corresponding to the vertices of *Lattice Mesh*. This step is shown in Figure 5.

Step 2

In this step, we generate curves corresponding to the lines C_0C_1 of *Lattice Mesh*. Define a vector between face point P_0 and edge point Q of M_1. (Edge point meaning a midpoint of the edge.) This vector represents the derivative vector of the curve. Start point P_0 and end point P_3 of the curve were calculated in Step1. Define the inner two control points P_1 and P_2 of the new curve. The two control points are calculated by the following functions:

$$P_0P_1 = \frac{4}{3} P_0Q$$

$$P_3P_2 = \frac{4}{3} P_3R$$

Now we can define a cubic Bézier curve with control points P_0,P_1,P_2,P_3. This step is shown in Figure 6.

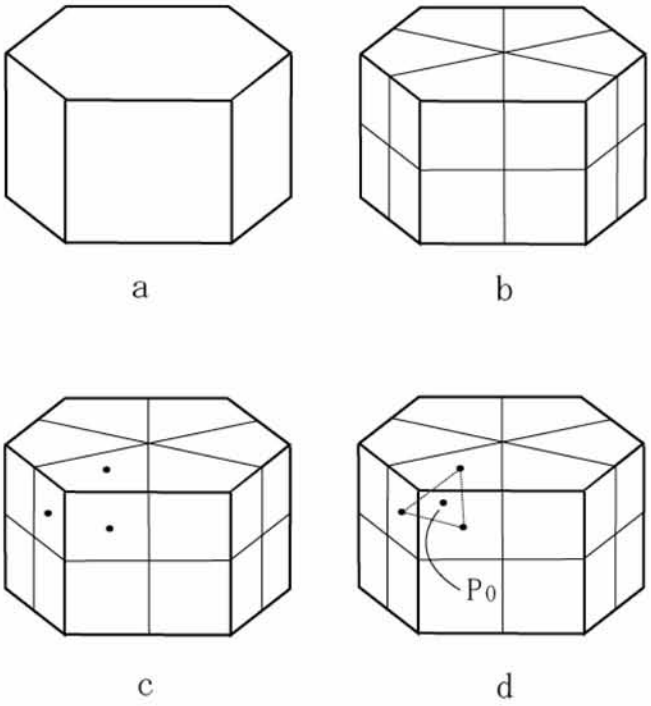

Figure 5: *Calculation of vertices on Lattice Surface. Initial Base mesh M_0(a). For each faces of M_0, connect face point and edge points(b). For each faces made in (b), calculate face points(c). Connect these face points and generate new Mesh M_1. For each faces of M_1, calculate face points P_1(d).*

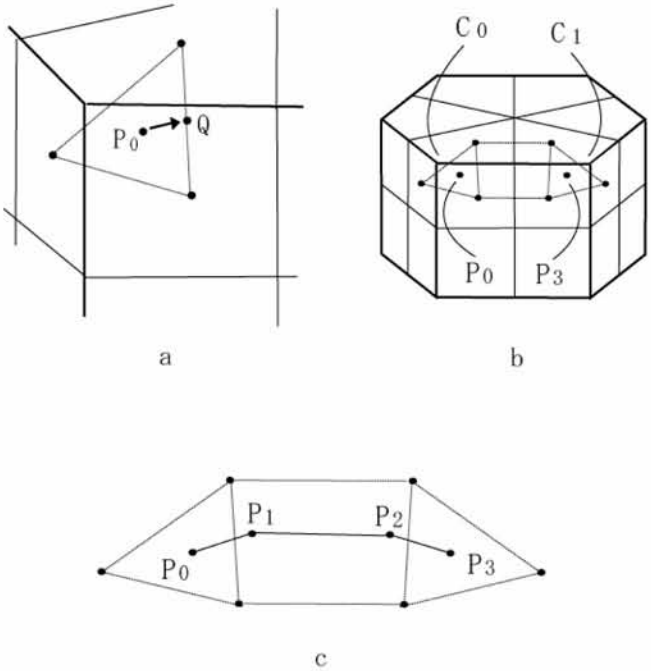

Figure 6: *Calculation of control points of the curve. Define a vector between face point P_0 and edge point Q of M_1 (a). P_0 and P_3 calculated in Step1 become start point and end point of the curve respectively (b). Define a Cubic Bézier curve with control points P_0,P_1,P_2,P_3 (c).*

Step 3

In the last step, using Chiyokura's method[3], the regions surrounded with cubic Bézier curves are interpolated by bicubic Gregory Patches. Generated *Lattice Surface* has the same topology as *Lattice Mesh*, as shown in Figure 1.

Rounding Weight

A wide range of surface shapes can be represented by specifying a weighting parameter to each edge and vertex. The main idea is to move the face points(P0 and P3) calculated in Step1. We can move these points by changing the filter of subdivision. This information is stored as weighting properties.

4. Applications with XVL

We have developed two applications to demonstrate the efficiency of using XVL. These are *Lattice Designer* and *Lattice Kernel*. *Lattice Kernel* is a solid modeling kernel, which is used as a base software to develop *Lattice Designer*, a solid based 3D design system.

4.1 Lattice Designer

Lattice Designer(Figure 7) is a 3D design system which offers intuitive user interface based on *Lattice Mesh* and XVL input/output. By transmitting 3D-Data with XVL, users can design collaboratively by sharing the same model on each screen.

The realistic human body model and MTB model, shown in Figure 8, are transferred as only 44KB and 39KB XVL data, respectively.

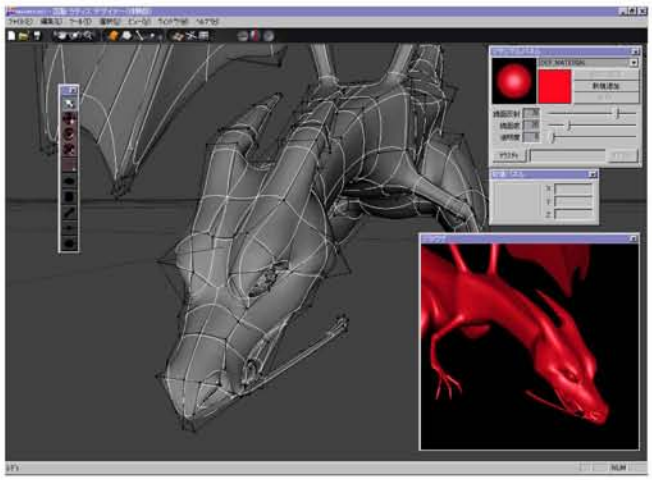

Figure 7: *Snapshot of Lattice Designer.*

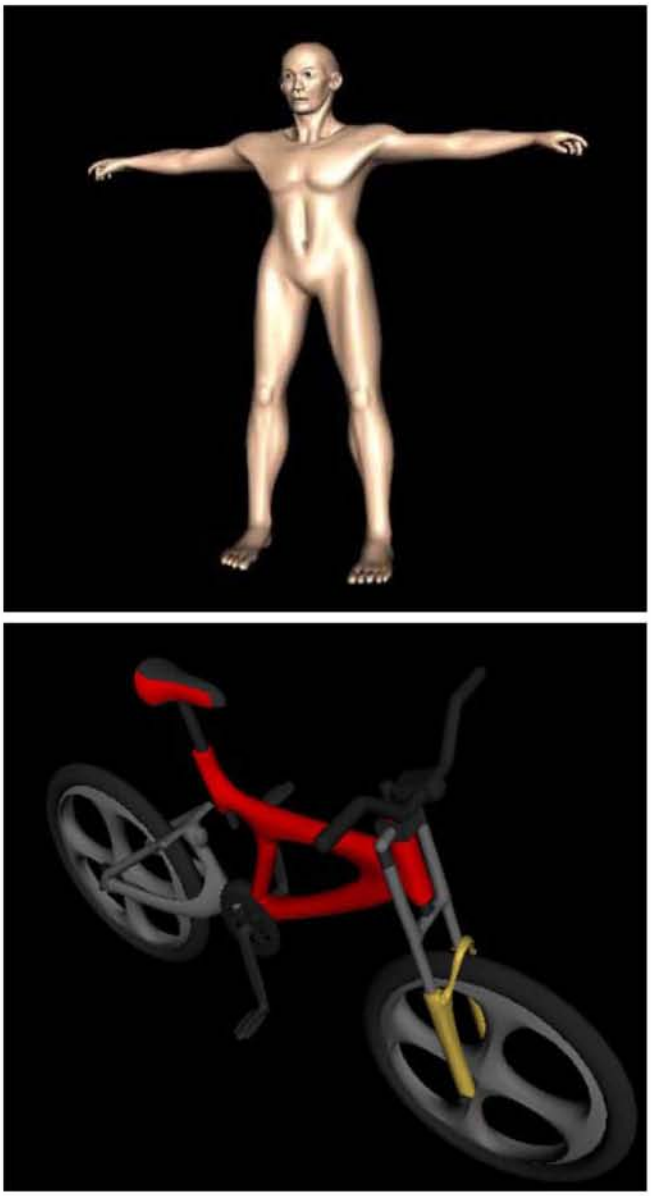

Figure 8: *Human body data(44KB) and MTB data(39KB)*

4.2 Surface Modeling with a Polygon Modeling-like Operation

Because *Lattice Surface* and *Lattice Mesh* are closely related, the user can create surface shapes by manipulating *Lattice Mesh*. This indicates that the user can create free-form surface shapes with a polygon modeling-like operation. Figure 9 shows an example of this practical modeling style.

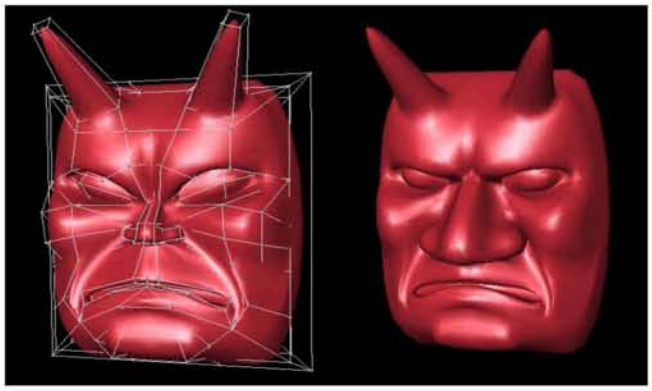

Figure 9: *Modeling with Lattice Mesh. User can create free-form surface shapes with polygon modeling-like operation.*

4.3 Dynamic Feedback Modeling

Simple modification of *Lattice Mesh* results in an immediate modification of *Lattice Surface*. This enables users to interactively change the surface shape by a type of real-time digital clay modeling. Figure 10 shows an example of this property.

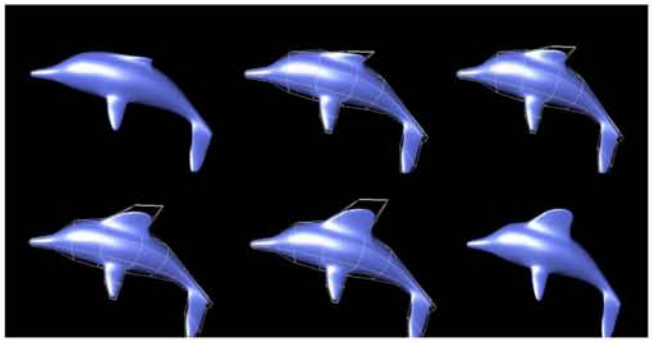

Figure 10: *Dynamic feedback modeling. Users can modify the surface shapes with real-time digital clay modeling.*

4.4 Direct Manipulation of a Surface

A direct modification of *Lattice Surface* is possible. Here, the corresponding meshes are internally re-generated by the algorithm described in Section 3. This enables a seamless operation of modeling the shape either by modeling a lattice or a surface.

4.5 Data Exchange with CAD/CAM Systems

Because *Lattice Designer* is a solid based modeling system, the data exported from *Lattice Designer* can be directly used in CAD/CAM systems. Moreover, XVL can be converted to STL format used in rapid prototyping (RP) systems. Generally, STL files are huge, therefore XVL offers a network solution to RP systems. Figure 11 shows a demon model made by a CAM system.

Figure 11: *Demon model made by CAM system.*

4.6 Lattice Kernel

In short, *Lattice Kernel* is developed as a scalable solid modeling toolkit transmitted in the XVL format. The application field may vary from CAD/CAM, to amusement, to 3D browsers. This type of singular toolkit was virtually impossible before. However, thanks to the scalability of the *Lattice Kernel*, users are now able to select only the required geometry and/or topology libraries of the kernel, thus minimizing the data size of the application program, as well as optimizing its function. Take a video game, for example. An application of the *Lattice Kernel* can be extremely compact, thus high-speed, by simply using the polygon module and omitting the complex surface.

5. Results

The table below provides evidence that indicates the performance of XVL for compact 3D-Data representation. Table 1 shows data size and processing time of XVL. The first column is the number of polygon mesh divided by *Lattice Surface* with an 8-division number. The second column is the data size of VRML(text). The third column is the data size of XVL(text). The fourth column is the time it takes to generate *Lattice Surface*. These data were obtained on a personal computer with PentiumIII 500MHz CPU.

Table1: *Performance of XVL.*

Data	result			
	ply num	VRML	XVL	surf gen (sec.)
hand	27126	1813KB	73KB	2.3
dolphin	24992	1647KB	128KB	2.1
raptor	68325	4343KB	343KB	3.4

6. Conclusions and Future Works

We have presented a framework of XVL as a compact 3D-Data format with high quality surface representation. The data structure can be migrated to VRML to use the Internet environment. Because XVL is based on a free-form surface, transferred XVL data can be directly utilized in CAD/CAM systems. This enables rapid transmission of practical data through the Internet. Lattice structure is composed of free-form surface and simple polygon mesh. These two data have the same topology and are interchangeable. A wide range of surface shapes can be generated using weighting factors. The rounding algorithm is so simple that rapid surface generation can be accomplished. Because of the compactness and ease of surface manipulation, XVL application fields are vast. Now we are extending XVL to migrate X3D. XVL is an extension of VRML, and therefore the same extension is easily accomplished in the framework of X3D. In order to completely exchange CAD/CAM data with *Lattice Kernel*, a future task is the support of trimmed-surface.

7. ACKNOWLEDGMENTS

The authors would like to thank Kazuhiro Taihei for modeling various sample data. Kunihiko Iijima, Hiroki Shibato and Seiji Takase for developing Lattice Designer. Souji Yamakawa for developing VRML toolkit.

8. REFERENCES

[1] V.Abadjev, M.del Rosario, A.Lebedev, A.Migdal and V.Paskhaver. *MetaStream*. VRML'99, 1999.

[2] E.Catmull and J.Clark. *Recursively generated B-spline surfaces on arbitrary topological meshes*. Computer Aided Design, pages 350-355, 1978.

[3] H.Chiyokura and F.Kimura. *Design of Solids with Free-form Surfaces*. Computer Graphics, pages 289-298, 1983.

[4] T.DeRose, M.Kass, and T.Truong. *Subdivision Surfaces in Character Animation*. Computer Graphics, 1998.

[5] D.Doo and M.Sabin. *Behaviour of Recurs ive Division Surfaces Near Extraordinary Points*. Computer Aided Design, pages 356-360, 1978.

[6] M.Eck, T.DeRose, T.Duchamp, H.Hoppe, M.Lounsbery, and W.Stuetzle. *Multiresolution Analysis of Arbitrary Meshes*. Computer Graphics, pages 173-182, 1995.

[7] H.Hoppe. *Progressive Meshes*. Computer Graphics, pages 99-108, 1996.

[8] M.Lounsbery, T.DeRose, and J.Warren. Multiresolution analysis for surfaces of arbitrary topological type. Technical Report 93-10-05b, Department of Computer Science and Engineering, University of Washington, January, 1994.

[9] G.Taubin, W.Horn, F.Lazarus, J.Rossignac, Proceedings of the IEEE, pages 1228-1243, vol. 96, no. 6, June 1998.

LATTICE TECHNOLOGY