Web-Based Tool for FRep Shape Modeling and Visualization

by

Mio Hiraga†

Faculty of Computer and Information Science
Hosei University
n00k1034@k.hosei.ac.jp

March 2004

† Supervisor: Alexander A. Pasko
Abstract

This paper presents a Web-based shape modeling and visualization tool with HyperFun programming language for FRep (function representation) models in the form of a Java applet. This Java applet is applicable as an online educational tool for students learning programming or computer graphics. Another way of using the applet is for collaborative shape modeling on the Web.

One of the main ideas of this paper is to develop a user friendly Web-based modeling system on the base of the existing tools provided: HyperFun program to the Java Bytecode Compiler, polygonization and the basic applet structure.

Keywords: Shape modeling, F-rep, Implicit Surface, HyperFun, collaborative modeling, Web applications, Java Applet, Java3D, Swing

The author is deeply grateful to Prof. Alexander Pasko for his kind instruction and encouragement, and to Yuichiro Goto for his technical help and support.

Also, I wish to thank my colleagues at our laboratory in Hosei University for all of their support.
Chapter 1

Introduction

Currently, the main types of three-dimensional (3D) modeling tools on the Web are Computer Aided Design (CAD) model browsers and converters such as [W97, RA99], virtual worlds authoring tools in Virtual Reality Modeling Language (VRML), and mathematical visualization tools for interactive geometry experiments such as [RM]. However, these tools are not pure shape modelers. In order to achieve on-line education and collaborative works for shape modeling of FRep solids [AV95], it is necessary to develop a new tool with interactive Graphical User Interface (GUI) as a pure shape modeler on the Web. In this paper, we present a “HyperFun Applet” as a new shape modeling tool with HyperFun [A99], which is a simple geometric modeling language for FRep solids, with a user-friendly GUI as a Java applet on the Web.

The prototype of the HyperFun applet [Y02] has already been developed for demonstration of Web-based shape modeling and visualization. The prototype applet in [Y02] included the following basic tools: HyperFun to Java Bytecode compiler, polygonizer and the basic applet structure that has an editor, viewer, error window, and a button. On the base of these tools, we present a new user-friendly modeler HyperFun Applet, which has optional menus for controlling rendering, supports saving the results in HyperFun, VRML, and STL formats at the client’s computer, and has a bilingual user interface.

The paper is structured as follows. First of all, the HyperFun language used as a modeling language of this tool is described in Chapter 2. The basic system of HyperFun Applet is described in Chapter 3. Chapter 4 presents the features of HyperFun Applet. Chapter 5 explains the implementation on the Web and discusses about problems in implementation. Conclusions and future works are discussed in Chapter 6.
Chapter 2

HyperFun Modeling Language

In this paper, we develop the Web-based tool for FRep shape modeling on the basis of HyperFun project [A99]. We introduce the project briefly in this chapter.

2.1. HyperFun project overview

The HyperFun project is a project devoted to developing open system architecture for functionally based shape modeling and its applications, as [A99] presents and also the HyperFun Project Web site (www.hyperfun.org) presents. The project is based on a function representation (FRep) [AV95] of geometric objects and supporting software tools built around the HyperFun language.

The project provides various modelers and viewers of HyperFun objects, and we will show the tools related on this thesis work.

- HyperFun for Windows (“HFW”) – This is an interactive system with GUI allowing for easy mastering FRep modeling concepts using the HyperFun language while working in a conventional MS Windows environment. It allows the user to specify the HyperFun model, to compose complex scenes with multiple objects, to specify parameters for controlling rendering and to generate animation.

- HyperFun Polygonizer (“HFP”) - This executable program polygonizes and displays an object from a HyperFun file in a conventional MS Windows environment. It has a command line interface allowing the user to define rendering options, and VRML output.

We aim for implementing more rendering options with interactive GUI in the prototype of HyperFun Applet, such as these tools have.
2.2. HyperFun modeling language

HyperFun [A99] is a very simple modeling language designed to be a tool suitable for specifying functionally based geometric models. The language was designed as simple as possible for ease of mastery. While being minimalist, the language does not limit the user in creating quite complex geometric models. It supports all main notions in FRep [AV95], particularly “geometric objects” and “geometric operations”. FRep defines a geometric object by a single real continuous function of several variables as:

\[ F(x_1, x_2, x_3, \ldots, x_n) \geq 0 \]

A model in HyperFun can contain the specification of several geometric objects. Each object is defined by a function parameterized by input arrays of point coordinates and free numerical parameters. The function can be represented with the help of assignment statements: conditional selection (‘if-then-else’) and iterative statements (‘while-loop’). Functional expressions are built using conventional arithmetic and relational operators. It is possible to use standard mathematical functions (‘exp’, ‘log’, ‘sqrt’, ‘sin’, etc.). Fundamental set-theoretic operations are supported by special built-in operators with reserved symbols (‘|’ – union, ‘&’ – intersection, ‘\’ – subtraction, ‘~’ – negation, ‘@’ – Cartesian product).

In principle, the language allows users to build objects from scratching FRep directly. However, it is available to use the system FRep library that contains functions representing geometric primitives and transformations, and the libraries available in the applet are shown in chapter 4.

In addition, the on-line manual for HyperFun language containing many examples of HyperFun models can be found in HyperFun Project Web site (www.hyperfun.org).
Chapter 3

HyperFun Applet System

3.1. HyperFun Applet system overview

The basic system of the HyperFun Applet can be divided into three important parts: HyperFun to Java Bytecode compiler, polygonizer and the applet interface, as shown in Fig. 3-1. These parts are practically implemented in seven packages: de, EmpiricalHyperFun, JaM2, java_cup, polygonizer, files and hyperfun_applet, as shown in Fig. 3-2.

Figure 3-1. HyperFun Applet System Overview
The four packages, “de, EmpiricalHyperFun, JaM2 and java_cup”, implement the work of HyperFun to Java Bytecode Compiler, and they have been implemented when the prototype of HyperFun Applet [Y02] was developed. The “polygonizer” package includes polygonizer class. The “files” includes HyperFun, VRML and STL files just for showing the name of them in the save dialog. The “hyperfun_applet” package includes the main class of applet and each class for exporting VRML and STL files.

3.2. HyperFun to Java Bytecode Compiler

HyperFun to Java Bytecode Compiler translates a HyperFun ASCII code into Java bytecode directly, and a class is synthesized from the bytecode on the memory. The class contains the method representing a FRep solid described in the HyperFun program given by the user. The synthesized class is loaded into the Java virtual machine using the dynamic class loading mechanism available in Java.

As a first step, a HyperFun to Java source code translator was developed as a part of a parsing courses taught at the University of Warwick in spring 1999, as noted in [EHF]. This translator required running the standard Java compiler to create the Java Bytecode from the HyperFun code as an external process to client computer. This is attributed to Java security model prevent an applet from the reading and writing local files, and from triggering external processes. To overcome these problems, the compiler was designed missing out the standard Java compiler completely and going directly from HyperFun to Java bytecode.
3.3. Polygonizer

Polygonization means a process of generating polygonal meshes from a functionally based surface model. The basic idea of polygonization [V88] takes two phases: spatial partitioning and cell polygonization. In spatial partitioning, the bounding box, \( x \in [x_1, x_2], y \in [y_1, y_2], z \in [z_1, z_2], \) as a modeling space of the shape is divided into cubic cells. Then, each cell is polygonized and the surface can be approximated with polygons.

Polygonization methods with using cubic cells have topological ambiguities on the faces with four edge-surface intersection points. The polygonizer of this applet was designed to solve the ambiguity problem and provide a fast polygonization method with the marching cubes algorithm [WH87] and the disambiguation algorithm described in [V88].

The package “polygonizer” implemented in this applet was developed by Y.Goto [Y02]. The package has Polygonizer class, and the class has some methods for generating polygonal mesh of HyperFun object and controlling rendering.

3.4. Applet interface

The screenshot of the HyperFun Java applet on the Web is shown in Fig. 4-1 of Chapter 4. The applet supports shape modeling work through the HyperFun to Java Bytecode compiler and the polygonizer, as noted above, and the visualized model depends on the modeling parameters obtained from the menus. Swing tools of Java 2 is used to implement the GUI, and Java 3D API is used to visualize the 3D mesh generated by polygonizer.

**Input data:** The input data for modeling are a HyperFun object in the form of HyperFun ASCII code, bounding box size, grid density, and surface color. These data are obtained from different parts of GUI and set by different parts of the system (Fig. 3-1), when the user clicks the “Polygonize” button.

The string data of the HyperFun code from the text area is set by the constructor of the Calculator class in the main program. Then a parser of HyperFun to Java Byte Code will start. The input data of Model Space and Grid Size menus will be
set using the methods of polygonizer class: void setBoundingBox() and void setDimensions(). The surface color data from JColorChooser will be set by the method in the main program: void setMaterial().

**Output data:** The output data are applet supports saving the results in HyperFun, VRML, and STL formats at the client’s computer. The HyperFun format is exported from the text area into the output window directly. The VRML format is exported from the vertices, normals and indices data generated by the polygonizer, and the color data from the color chooser by using vrmlOut class. The STL format can be exported from the vertices and indices data generated by the polygonizer. In addition, the STL format needs facet normal data, so we calculate normals by the cross product calculation for each facet in stlOut class.
3.5. Interactive modeling process

The interactive modeling process with the applet can be described as below Fig. 3-3.

Figure 3-3. Flowchart of the interactive modeling process with the applet
3.6. Export process

The applet supports saving the final HyperFun model and the triangular mesh in VRML and STL formats at the client’s computer. We describe the processes of exporting VRML and STL format from HyperFun model in this sub-section.

3.6.1. VRML export process: VRML (Virtual Reality Modeling Language) is an open language that can create 3D space on the Web, and it is the global standard format for which does not depend on platform. VRML builds the 3D world by defining the object that includes various data, such as 3D shapes, sound data and picture data, and the object is defined as a node. In this export process, we define the HyperFun object given by the user as a Shape node in VRML2.0.

In VRML format, a 3D polyhedral shape can be described as a group of polygons, and its geometry can be described as an IndexedFaceSet node included in a Shape node. The IndexedFaceSet node needs to include each field of vertices coordinates, normal vectors, coordinate indices, and normal indices, and the coordinates are listed in counterclockwise order when looking at the object from the outside (right-hand rule). For example, three faces shown in Fig.3-4 (a) can be defined as Fig.3-4 (b). Note that the arrangement of the coordinate indices has a characteristic that “-1” means the end mark of a definition of one face.

![Figure 3-4](image)

(a) (b)

**Figure 3-4.** The description of polygons in VRML, (a) the polygons, (b) the example of arrangement of the coordinates and their indices
The HyperFun Applet exports the VRML file from the HyperFun object using hyperfun_applet.VrmlOut class, as follow process.

1. “Save as VRML” button in the save menu is pushed after polygonization.
2. An instance of VrmlOut class in the “hyperfun_applet” package will be generated, and the constructor of the VrmlOut class imports the arguments: color data received from the color chooser, vertices, normals and indices generated by the polygonizer.
3. In the method “public String vrmfout()” in VrmlOut class, string data in the VRML format will be generated from the arguments, and string data returns to the main program.
4. The string data including the VRML data will show on the output window, using the showOutputWindow() method in the main program.
5. When “Save” button on the output window is clicked, the save dialog of JFileChooser will show and save the default file “output.wrl” on the client’s machine.

3.6.2. STL export process: An STL (Stereo Lithography) file format is a standard data format for rapid prototyping machine, which makes physical 3D solid objects automatically from digital data.

The STL format is an ASCII or binary file, and the format is a triangular representation of a 3D surface geometry. A triangle is called as facet, and a STL file consists of a list of facet data. Each facet is uniquely identified by a unit normal with a length of 1.0 and by three vertices, as shown in Fig 3-5.

![Figure 3-5. The description of STL format, (a) a facet, (b) STL data format](image)
In order to export a STL file from a HyperFun object, it needs to calculate each unit normal vector of a facet. A unit normal vector can find the results of cross product calculation on each facet (Fig.3-6) as below.

\[
A = P3 - P1 = (x_3 - x_1, y_3 - y_1, z_3 - z_1) \\
B = P2 - P1 = (x_2 - x_1, y_2 - y_1, z_2 - z_1) \\
A \times B = (Ay \cdot Bz - Az \cdot By, Az \cdot Bx - Ax \cdot Bz, Ax \cdot By - Ay \cdot Bx) \\
UnitNormalVector = (Nx, Ny, Nz) = \frac{A \times B}{|A \times B|}
\]

**Figure 3-6.** The vectors for calculating a unit normal vector on a facet

The HyperFun Applet exports the STL ASCII file from the HyperFun object using hyperfun_applet.StlOut class, as follow process.

1. “Save as STL” button in the save menu is pushed after polygonization.
2. An instance of StlOut class in the “hyperfun_applet” package will be generated, and the constructor of the StlOut class imports the arguments: vertices and indices generated by the polygonizer.
3. In the method “public String stlout()” in StlOut class, string data in the STL ASCII format will be generated from the vertices list on the base of indices, and facet normal calculation described as above.
4. The string data in STL ASCII format returns to the main program.
5. The string data including the STL data will show on the output window, using the showOutputWindow() method in the main program.
Chapter 4
Applet Features

In this section, we will present the features of HyperFun Applet. The interactive interface of the applet through the browser is shown in below Fig. 4-1.

Figure 4-1. HyperFun Applet Interface on the browser
4.1. Modeling menus

**HyperFun program editor:** First of all process, the user edits a HyperFun program on the HyperFun program editor, shown in Fig. 4-1(1). It has scroll bar when the text is long.

**Polygonize button:** When the user starts to rendering after editing the HyperFun program, he/she just click the “POLYGONIZE” button in Fig. 4-1(5).

**Error message window:** The text area in Fig. 4-1(3) is provided for showing error messages. If the HyperFun program has syntax errors, the user can see the messages of the errors in this area such as “Syntax error. Couldn't repair and continue parse”. If the user is going to save the model as VRML or STL format before polygonization, the error message “Please save the file after POLYGONIZE !”

**Graphics window:** The HyperFun model polygonized by the applet is visualized by Java 3D in the graphics window, shown in Fig. 4-1(2). The graphics window shows the HyperFun model, the bounding box and xyz-axis. The bounding box is shown in White lines. X-axis is shown in a Red line, Y-axis is shown in a Green line, and Z-axis is shown in a Blue line. In addition to viewing the 3D model, the user can rotate by left click, shift by right click, and zoom by center click.

4.2. Controlling rendering menus

The applet has optional menus for controlling rendering the HyperFun object in the middle of right side on the applet, shown in Fig. 4-1(6). These menus are provided in the form of tab menu.
**Model space:** In “Model Space” menu, shown in Fig. 4-2, the user can input the value of the bounding box size as a model space for his/her HyperFun model. The user needs to input the minimum point (Xmin, Ymin, Zmin) and the maximum point (Xmax, Ymax and Zmax).

![Model Space Menu](image)

**Figure 4-2. Model Space Menu**

For example, a sphere (radius = 3) is visualized in different bounding boxes as below Fig. 4-3. The default of the bounding box, shown in Fig. 4-3(a), is a cube where each edge length is 10. In the left one (a), the bounding box completely encloses the sphere. In the right one (b), the sphere has holes, because the outside of the bounding box was not calculated.

![Sphere models in different bounding box](image)

(a) Edge length = 10
Min (-5,-5,-5)  
Max (5, 5, 5)

(b) Edge length = 5
Min (-2.5,-2.5,-2.5)  
Max (2.5, 2.5, 2.5)

**Figure 4-3. Sphere models in different bounding box**
**Grid:** In “Grid” menu, shown in Fig. 4-4, the user can change the grid density in the bounding box. The meanings of Grid X, Grid Y and Grid Z in the menu are the density of the density of x, the density of y and the density of z, respectively. The default value is Grid X = 30, Grid Y = 30, Grid Z = 30.

In default value, the bounding box will be divided into 30*30*30 cells, and then the polygonizer will generate polygons in each cell. The higher density increases the number of polygons, accuracy higher, but it takes time to calculate. The time depends on the memory available on user’s computer.

![Figure 4-4. Grid Menu](image)

For example, a torus as hfTorusZ(x, center, 3, 1.5) is visualized in the different grid density as below Fig. 4-5.

![Figure 4-5. Torus models in different grid density](image)
**Color:** In “Color” menu, shown in Fig. 4-6(a), the user can select the color of the HyperFun object. If the user clicks the select button in color menu, the color dialog will appear and user can select a color from sample Fig. 4-6(b), HSB Fig. 4-6(c) or RGB Fig. 4-6(d).

- Sample — for choosing a color from a collection of swatches.
- HSB — for choosing a color using the Hue-Saturation-Brightness color model.
- RGB — for choosing a color using the Red-Green-Blue color model.

For example, a castle model is visualized in the different color as below Fig. 4-7. Fig. 4-7 (a) is the model in default color R=0, G=204, B=51 (H=135, S=100, B=80), and Fig. 4-7 (b) is selected color from Sample equal to R=255, G=51, B=255 (H=299, S=80, B=100).
In “Library” menu, the user can reference predefined FRep library functions available in HyperFun, which is produced by HyperFun Project [A99]. The functions in below Table 4-1 are available at this writing.

**Table 4-1. Available FRep library Functions in the HyperFun Applet**

<table>
<thead>
<tr>
<th>Primitives</th>
<th>Oprations</th>
</tr>
</thead>
<tbody>
<tr>
<td>hfSphere(x, center, R);</td>
<td>hfScale3D(xt, sx, sy, sz);</td>
</tr>
<tr>
<td>hfEllipsoid(x, center, a, b, c);</td>
<td>hfShift3D(xt, dx, dy, dz);</td>
</tr>
<tr>
<td>hfCylinderX(x, center,R);</td>
<td>hfRotate3DY(xt, theta);</td>
</tr>
<tr>
<td>hfCylinderY(x, center,R);</td>
<td>hfRotate3DZ(xt, theta);</td>
</tr>
<tr>
<td>hfCylinderZ(x, center,R);</td>
<td>hfRotate3DZ(xt, theta);</td>
</tr>
<tr>
<td>hfEllCylX(x, center, a, b);</td>
<td>hfBlendlntr(f1,f2,a0,a1,a2);</td>
</tr>
<tr>
<td>hfEllCylY(x, center, a, b);</td>
<td>hfTwistX(xt, x1, x2, theta1, theta2);</td>
</tr>
<tr>
<td>hfEllCylZ(x, center, a, b);</td>
<td>hfTwistY(xt, y1, y2, theta1, theta2);</td>
</tr>
<tr>
<td>hfTorusX(x, center, R, r0);</td>
<td>hfTwistZ(xt, z1, z2, theta1, theta2);</td>
</tr>
<tr>
<td>hfTorusY(x, center, R, r0);</td>
<td>hfStretch3D(xt,x0,sx,sy,sz);</td>
</tr>
<tr>
<td>hfTorusZ(x, center, R, r0);</td>
<td>hfTaperX(xt, x1, x2, s1, s2);</td>
</tr>
<tr>
<td>hfBlock(x, vertex, dx, dy, dz);</td>
<td>hfTaperY(xt, y1, y2, s1, s2);</td>
</tr>
<tr>
<td>hfBlobby(x,x0,y0,z0,a,b,T);</td>
<td>hfTaperZ(xt, z1, z2, s1, s2);</td>
</tr>
<tr>
<td>hfMetaBall(x,x0,y0,z0,b,d,T);</td>
<td></td>
</tr>
<tr>
<td>hfSoft(x,x0,y0,z0,d,T);</td>
<td></td>
</tr>
</tbody>
</table>
4.3. Saving files

The applet can export files, in order to be applicable for not only just a test on the Web but also continual or collaborative work. HyperFun is a specialized high-level programming language for specifying F-rep models, but this language is not popular so much such as the standard modeling language like VRML. Therefore, we prepared conversion a HyperFun model into the other two formats: VRML and STL ASCII Format.

In “Save” menu, shown in Fig. 4-8, the user can select the format to save. When user pushes each button, the output window Fig.4-9 will appear and show the output data on the text area. Until this process, the applet does not load or write files on the client computer. If the user pushes the “Save” button in the upper on the output window, the save dialog will appears. The default name of each format are “output.hf”, “output.wrl” and “output.stl”. The each file type description of the format selected by the user is shown in the file type on the dialog. When the save button is pushed, the file is saved as the selected file name on the selected directory.

![Figure 4-8. Save Menu](image1)
![Figure 4-9. Output window and Save dialog](image2)
**HyperFun File Format:** The applet can save the HyperFun object as HyperFun data file, which is a text file as “.hf” extension. Fig. 4-10 shows an object exported by the HyperFun Applet, which is visualized by the HyperFun for Windows (HFW) system.

![HyperFun model output visualized by HFW](image)

**Figure 4-10.** HyperFun model output visualized by HFW

**VRML File Format:** VRML (Virtual Reality Modeling Language) is an open language specification that helps create 3D worlds on the Web, and it is a global standard format which does not depend on platforms similar to HTML.

This applet can convert the HyperFun object into the VRML v. 2.0 format and save it on the client’s computer. This file needs to be saved with the “.wrl” extension and it can be visualized through a browser or a Web browser plug-in. In Fig. 4-11, we show a VRML file exported to VRML from the same HyperFun object as in Fig. 4-10 and visualized by Cosmo Player, which is a premiere VRML viewer for Web browsers.
STL File Format: STL (Stereo Lithography) is a standard data format for rapid prototyping machine, which makes 3D solid objects automatically from digital data. The STL format is a triangular representation of a 3D surface geometry. This applet can convert the HyperFun model into the STL ASCII format. This file needs to be saved with the "*.stl" extension. We show in Fig. 4-12 an STL file exported from the same HyperFun example as in section Fig. 4-10 and visualized by VisCam RP, which is the software package for the complete preparation of CAD/CAM data for any rapid prototyping application.

Figure 4-11. VRML output visualized by Cosmo Player

Figure 4-12. STL ASCII output visualized by the VisCam RP rapid prototyping software
4.4. Bilingual interface

This applet shows menus in a bilingual form of English and Japanese. It can change the language of menu by pushing the button in the lower right side, shown in Fig. 4-1(4). In Japanese mode, the language button appears as “English”. In English mode, the language button appears as “Japanese”.
Chapter 5

Implementation

5.1. System requirements and testing

The applet needs to have been installed “Java 2 JRE or SDK” and “Java 3D” for running on your browser on your any machine. We have developed and run the applet with Java(TM) 2 SDK, Standard Edition Version 1.4.2 and Java 3D(TM) 1.3.1 on Microsoft Windows 2000 [Version 5.00.2195] Service Pack 4. In addition, we had tested and confirmed this applet available to run on below browsers.

<table>
<thead>
<tr>
<th>Platform</th>
<th>OS</th>
<th>Browser</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>2000, XP</td>
<td>InternetExplorer6 NetScape 7</td>
<td>J2RE or J2SDK + Java3D (OpenGL or DirectX)</td>
</tr>
<tr>
<td>Linux</td>
<td>TurboLinux</td>
<td>Mozilla</td>
<td>J2SDK + Mesa3D (OpenGL) +Java3D for Linux</td>
</tr>
<tr>
<td>Macintosh</td>
<td>OSX10.3.1</td>
<td>Safari1.1</td>
<td>Java3D</td>
</tr>
</tbody>
</table>

5.2. Implementation on the Web

In general, untrusted Java applets loaded over the net are prevented from reading and writing files on the client file system because of its security problem, such as unauthorized access or damage to data. When we upload our HyperFun applet on the Web, the applet needs to be considered as a trusted applet, because the applet accesses the hard disk on the client machine in writing files: HyperFun files, VRML files and STL files. In order to achieve considering the applet to be trusted, we signed the applet by below process using Java Archive tool and keytool included in Java 2 SDK.
1. Package all files into a JAR (Java Archive) file.
   \% jar cf  hfapplet.jar [ files ]

2. Generate a key by using “keytool” and input the information of developer.
   \% keytool –genkey –alias [ keyname ]

3. Sign on the JAR file by the generated key
   \%jarsigner hfapplet.jar [ keyname ]

5.3. Implementation problems

When a HyperFun model edited by the user has any kind of errors, always the message “Syntax error. Couldn't repair and continue parse” is shown in the error message area of the applet. The problem that the applet has no other kind of error messages makes user very uncomfortable, because user has no way to know where he had made the error.

The reason of showing only the unfriendly error message concerns the HyperFun to Java compiler implemented when the prototype of the applet was developed [Y02]. The compiler was implemented using Jlex and Java Cup tools, which are parser generators. The compiler was designed to translate HyperFun to Java Bytecode directly, so it doesn’t handle the errors in the intermediate process. Therefore, one of the methods to handle the errors is to implement a simple parser for only syntax error checking, which works before the compiler to Java Bytecode process starts. To implement the simple parser, we consider two methods.

The first method is to make a simple parser using not JavaCup, but JavaCC (Java Compiler Compiler). Contrary to Java Cup provides a one type error reporting such as mentioned above, JavaCC provides very good error reporting, which include error token, line number, column number and expecting words, when there are syntax errors in parsing. Note that JavaCC disallows left-recursion in the grammar, thus the grammar of HyperFun needs to be removed left-recursions. It can be solved by rewriting the grammar in the form of no left-recursions by below algorithm.
Besides, JavaCC allows to extended BNF (Backus Naur Form) specifications, such as \((A)^*\) and \((A)^+\), within the lexical and the grammar specifications. Below expression will also solve the left-recursion problem.

\[
A \rightarrow A\alpha|\beta \quad \text{\RedRightarrow} \quad A \rightarrow \beta A' \\
A' \rightarrow \alpha A'|\epsilon
\]

The second method is to develop a parser in Java for processing a HyperFun program one by one character on the base of the HyperFun grammar. In this case, the HyperFun grammar is LL(1), and the parser can be constructed a top-down or recursive descent parser in Java.
Chapter 6

Conclusions and Future Works

In this paper, we extended a FRep shape modeling and visualization tool, “HyperFun Applet”, which has GUI for controlling rendering, supports saving the results in HyperFun, VRML, and STL formats at the client’s computer, and has bilingual user interface.

The simple modeling language for FRep shape, HyperFun, is introduced in Chapter 2. It can be exchange protocol of representing shape on the Web.

The interactive modeling system of the HyperFun Applet is showed in Chapter 3. The new features added on the prototype of HyperFun applet was shown in Chapter 4. The easiness of controlling rendering helps users to understand the rendering principles. In addition, saving the HyperFun model in other formats makes it a really applicable modeler connected to other standards. The VRML output will make it possible to add HyperFun models to a graphical scene supported in VRML and not covered in HyperFun itself. The STL output makes it possible to produce physical 3D objects using rapid prototyping equipment.

In Chapter 5, the running test on Windows, Linux and Macintosh platform showed the applet is platform independent tool. It means the applet is available on the any computer all over the world.

In order to achieve more useful “HyperFun Applet”, it is needed to implement more detailed syntax error checking as mentioned in section 5.3. In addition, HyperFun can describe animation using the dynamic coordinates representing continuous values, as [A99] mentioned. So, tools for generation animations are also needed.
References


