Design and Implementation of Animation Post-processor Based on ACIS and HOOPS in MWorks

Zhou Fanli[1], Zhang Hehua[2], Zhu Hengwei[2], Gong Xiong[1], Wang Boxing[1], Liu Jun[1], Chen Liping[1], Huang Zhengdong[1]
[1]Huazhong Univ. of Sci.&Tech., CAD Center, Wuhan, China
[2]Suzhou Toprank Software & Control Tech. Co. Ltd, Suzhou, China
{fanli.zhou, zhanghehuahust}@gmail.com, zhuhwei@126.com, {gongx, wangbx, liuj, chenlp}@hustcad.com, zdhuang@hust.edu.cn

Abstract

A complete Modelica-based simulation platform usually consists of modeling tool, compiler, analyzer, solver and post-processor. The 3D animation function is essential to the post-processor of a platform that supports MultiBody system simulation. Taking advantage of the complementarity and interoperability between graphical engines ACIS and HOOPS, MWorks, as a new generation of multi-domain modeling and simulation platform, implements the 3D animation of its post-processor based on these two graphical engines, and provides plentiful animation functions.

This paper firstly presents the overall design of the animation post-processor based on the analysis of visual features of the standard multibody library in Modelica; then describes its implementation, including mechanisms of geometry creation and display, data management and interactive interface; finally, verifies the effectiveness of the post-processor by some typical examples from the multibody library and application to aircraft landing gear simulation.

Keywords: Modelica; Post-processor; 3D animation; ACIS & HOOPS; MWorks

1 Introduction

A Modelica-based simulator usually consists of modeling tool, compiler, analyzer, solver and post-processor. The basic function of post-processor is to display simulation results in curves. If a platform supports multibody system, the 3D animation function is essential to its post-processor. The animation post-processor is used to deal with multibody animation, including geometry creation, graphic rendering, animation control and so on.

The popular graphic engines include PARASOLID, OpenGL, ACIS[1], HOOPS[2], Granite, etc. None of them has complete functions in animation. PARASOLID is good at modeling and visual interaction but has a defect in data management of complex models due to its unclear data structure; OpenGL has powerful graphical display and interaction functions but is short of professional geometric library; ACIS provides plentiful geometry modeling functions but is weak in visual operation and interaction; HOOPS has significant advantages in graphical display, interaction and data structure but is not good at modeling. Therefore, it's difficult to develop a powerful graphic system based on only one graphical engine.

Some simulation platforms provide animation function for multibody systems based on VRML, but this method is not powerful enough due to its defects in graphical quality, kernel interfaces and geometry library. Taking advantage of the complementarity and interoperability between ACIS and HOOPS, MWorks, as a new generation of multi-domain modeling and simulation platforms, implements the 3D animation of its post-processor based on these two graphical engines. MWorks provides plentiful animation functions, which have the advantages of convenient human-computer interaction, good geometric format compatibility, real-time geometric rendering, high fidelity animation effects, powerful model management and high expandability.

2 Design Overview

2.1 Visual Features of Standard MultiBody Library in Modelica

The standard MultiBody library in Modelica 2.2.2 or later consists of packages of World, Examples, Forces, Frames, Interfaces, Joints, Parts, Sensors, Types and Visualizers, as shown in Figure 2.1.

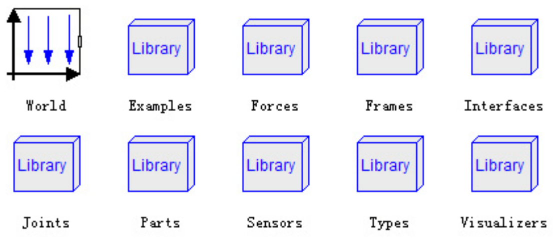


Figure 2.1 Modelica Standard MultiBody Library

Visualizers is the 3D graphic visualization package of the MultiBody library, which includes models of FixedShape, FixedShape2, FixedFrame, FixedArrow, SignalArrow, Advanced.Arrow, Advanced.Double-Arrow, Advanced.Shape, Internal.FixedLines and Internal.Lines. The Advanced.Shape model is the core of the Visualizers package, which gives the information about geometry construction in multibody animation.

The geometry is created according to 7 output variables in Visualizers.Advanced.Shape model, which are Form, rxvisobj[3], ryvisobj[3], rvisobj[3], size[3], Material and Extra. The variable Form represents the shape of multibody part, which may have two types: one is from the eight basic geometric elements in the standard library: box, sphere, cylinder, cone, pipe, beam, gearwheel and spring (see Figure 2.2); the other is from imported geometries defined by external geometric files, which have no unified format. The variables of rxvisobj[3], ryvisobj[3] and rvisobj[3] specify the position of model relative to the world coordinate system. The variable size[3] describes the length, width and height of model as shown in Figure 2.2, in which the dark blue arrow means the length direction and the light blue arrow means the width direction. The variable Material depicts material properties of model including color and specular coefficient. The variable Extra implies additional graphic properties, which have different meanings for different elements, as shown in Table 2.1.

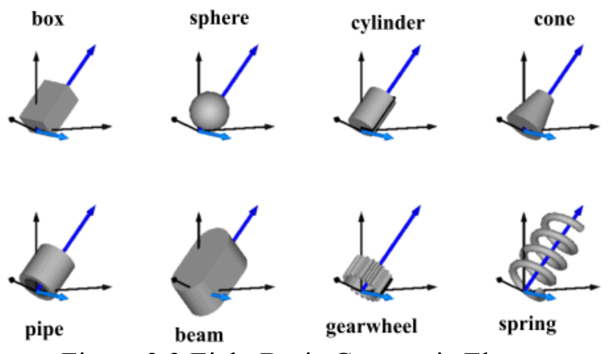


Figure 2.2 Eight Basic Geometric Elements

Table 2.1 Meaning of Variable Extra

Shape Type	Meaning of Variable Extra
cylinder	If Extra > 0, a black line is included in the cylinder to show its rotation.
cone	Extra = diameter-left-side / diameter-right-side, i.e; Extra = 1: cylinder; Extra = 0: “real” cone.
pipe	Extra = outer-diameter / inner-diameter, i.e; Extra = 1: cylinder that is completely hollow; Extra = 0: cylinder without a hole.
gearwheel	Extra is the number of teeth of the gear.
spring	Extra is the number of windings of the spring. Additionally, “height” is not the “height” but 2*coil–width.

The geometry of every multibody part is an assembly of different instances of the Visualizers.Advanced.Shape model. As an example, the instances of Shape in the example model Modelica.Mechanics.Examples.Elementary.DoublePendulum are shown in XML file in Figure 2.3.

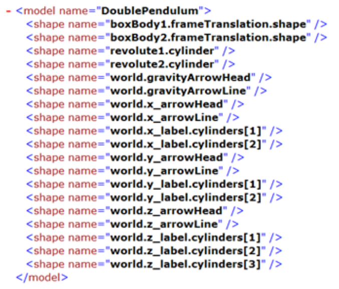
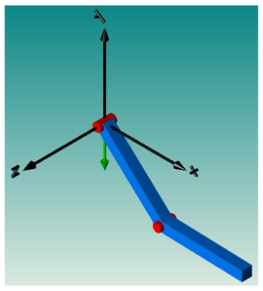

```
- <model name="DoublePendulum">
    <shape name="boxBody1.frameTranslation.shape" />
    <shape name="boxBody2.frameTranslation.shape" />
    <shape name="revolute1.cylinder" />
    <shape name="revolute2.cylinder" />
    <shape name="world.gravityArrowHead" />
    <shape name="world.gravityArrowLine" />
    <shape name="world.x_arrowHead" />
    <shape name="world.x_arrowLine" />
    <shape name="world.x_label.cylinders[1]" />
    <shape name="world.x_label.cylinders[2]" />
    <shape name="world.y_arrowHead" />
    <shape name="world.y_arrowLine" />
    <shape name="world.y_label.cylinders[1]" />
    <shape name="world.y_label.cylinders[2]" />
    <shape name="world.z_arrowHead" />
    <shape name="world.z_arrowLine" />
    <shape name="world.z_label.cylinders[1]" />
    <shape name="world.z_label.cylinders[2]" />
    <shape name="world.z_label.cylinders[3]" />
  </model>
```


Figure 2.3 Geometries of DoublePendulum

2.2 Framework

MWorks[5][6] consists of five modules: Modeling environment, Compiler, Analyzer, Solver and Postprocessor. Modeling environment allows users to new a Modelica model by using drag-drop operation or text. Compiler compiles models by running lexical, syntax and semantic checks and generates flat equation systems of models. Analyzer analyzes flat equation systems from Compiler by carrying out structural consistency check, variable substitution, BLT decomposition and high index DAE reduction, and outputs index-1 DAE equation sequences. Solver solves the index-1 DAE equations in order and outputs the file of simulation results. Post-processor reads the result file and displays results in curves or in animation.

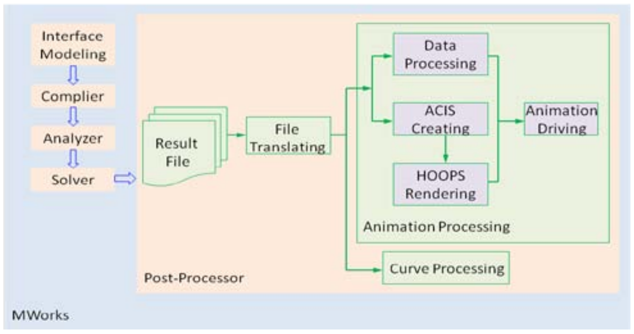


Figure 2.4 Process of MWorks

The process of animation post-processor in MWorks is as follows (see Figure 2.4): Firstly, post-processor reads and parses the result file to generate information for animation including geometric data (shape, position, material), animation data, curve data, etc.; Secondly, the post-processor uses ACIS to create geometries based on geometric data, and then uses HOOPS to render and display 3D geometric models; Thirdly, the post-processor generates transformation matrices of each animation frame based on animation data, which drive model to move; Finally, the post-processor responds to user's operations to begin animation, stop animation, rotate or translate model and so on.

3 Implementation

The key factors of the implementation of the animation post-processor include process of geometry creation and display, performance of data management and convenience of interactive interfaces.

3.1 Geometry Creation and Display

The mechanism of geometry creation and display is the core of animation post-processor, and its design directly affects the performance of the post-processor. The process of creating and displaying geometry in MWorks is shown below (see Figure 3.1):

(1) Create a top geometric model and initialize it;
(2) Check whether all parts have been created, if yes, go to step (6), if no, go to step (3);
(3) Create a part in the top model relative to the world coordinate system;
(4) Create a geometric entity relative to the part coordinate system by the following steps:

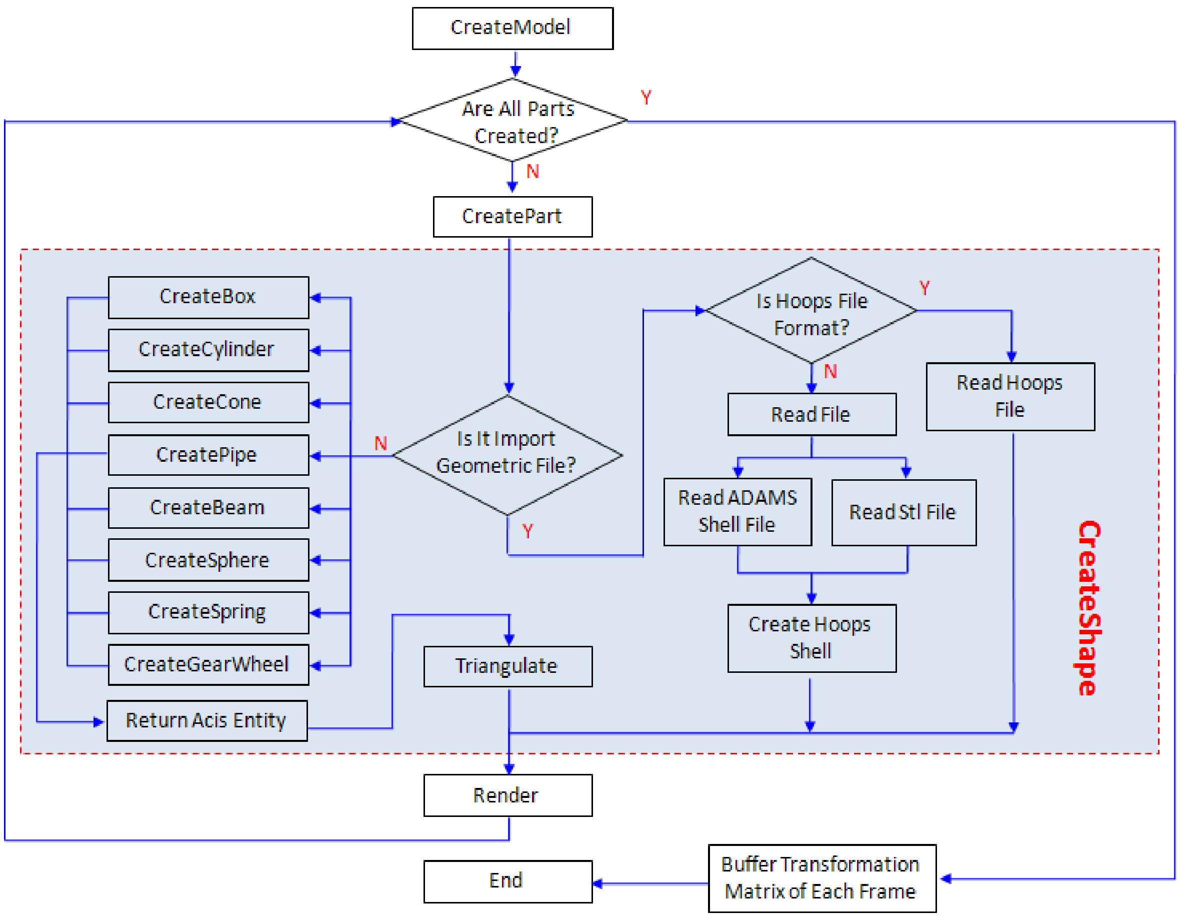


Figure 3.1 Process of Geometry Creation and Display

(i) Check whether the geometry is defined by an externally imported graphic file, if no, that is, the geometry is a standard graphic element, go to step (ii); if yes, go to step (iii);

(ii) Invoke the responding ACIS APIs to create geometric entity according to the element type of the geometry, then triangulate it and generate HOOPS Shell (a collection of polygons that forms a 3D object);

(iii) Check whether the imported file is in HOOPS format, if no, invoke self-defined APIs to parse the file and generate HOOPS Shell; if yes, invoke HOOPS APIs to parse the file and return the geometric object;

(5) Render the geometry, then go to step (2);

(6) Read the result file to generate transform matrices of each animation frame and save them to buffer;

(7) Drive animation of the multibody model.

At present, the post-processor of MWorks can support the following formats: STL file (.stl), HOOPS file (.hsf and .hmf), ADAMS shell file (.shl), etc.

3.2 Data Management

3.2.1 Management of Geometric Data

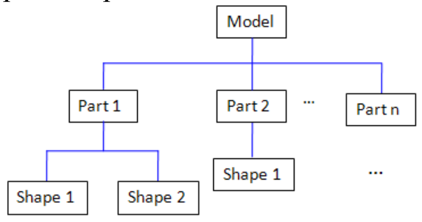
After reading the result file, the post-processor obtains the data used for creating geometries. In order to improve the efficiency of accessing data, the geometric data of all instances of the Visualizers.Advanced.Shape model are saved in special data structure combining map container and struct pointer. The definition of data structure is given below:

```
struct GeoDataStruct
{
   string shapeType;                  // shape
   double size[3];                    // {length, width, height}
   double extra;
   double specularCoefficient;
   unsigned char color[3];            // RGB
   MbsFrame frame;                    // {rvisobj[3], rxvisobj[3], ryvisobj[3]}
};
map<string/*full name of shape*/, GeoDataStruct*>  mapGeoDataStrcut;
```

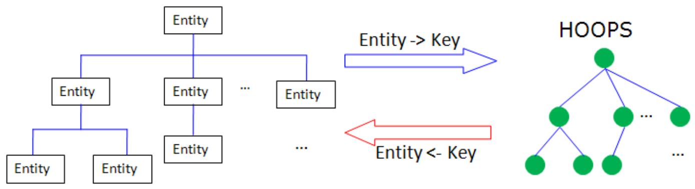
3.2.2 Management of Model Data

The post-processor of MWorks uses tree structure to represent model data. A Model, which represents a multibody model, contains a number of Parts. A Part consists of a number of Shapes, which implies an instance of Visualizers.Advanced.Shape model (see Figure 3.2). Meanwhile, HOOPS uses Segment to describe model data, and one segment maps one HOOPS key. So the key problem in management of model data is how to build the tree structure of model using the HOOPS key.

MWorks uses C++ inheritance mechanism to build the two-way mapping between Entity pointer and HOOPS key by creating Entity class (all of Model, Part and Shape are inherited from Entity). This method can effectively solve the key problem in management of model data. We can use the implementation of highlight picking up as an example: we firstly use mouse to select certain geometric object, and then invoke HOOPS API to obtain the HOOPS key of that object. The corresponding entity pointer of that object can be obtained by the two-way mapping. We finally invoke interfaces to modify the color and transparency of the entity, which indicates that the object is picked up.



Model Hierarchy



Two-Way Mapping between Entity and HOOPS Key

Figure 3.2 Structures of Model Data

3.2.3 Management of Animation Data

The 3D animation can be viewed as display of a sequence of picture frames. The position of each part of model has been changed once after each picture frame is displayed, which can be represented by a 4*4 matrix, namely transformation matrix. In order to enhance the efficiency of reading and writing animation data, we adopt the consecutive memory storage method (see Figure 3.3). This method stores the data of the same type in a continuous memory area, so that the data can be easily accessed through its first address and block length.

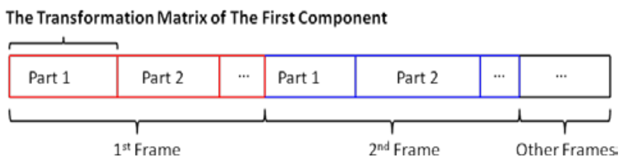


Figure 3.3 Physical Structure of Animation Data

3.3 Interactive Interfaces

The animation post-processor of MWorks uses MFC multiple document/view framework, which allows the user to open a number of relatively independent animation windows at the same time. The animation interface menu provides four kinds of functions (see Figure 3.4): (1) view splitting function to allow users to view animation in different split views from different perspectives; (2) model operation function to allow users to rotate, translate, zoom and highlight pick up model and to change the display mode, which can be wire-frame mode, hidden mode, perspective mode and shadow mode; (3) view operation function to allow users to change the observing view, which can be front view, rear view, left view, right view, upward view, downward view or axonometric view, etc.; (4) animation control function including the operations of starting animation, suspending animation, reversing animation, resetting animation, adjusting animation speed or recording animation video.

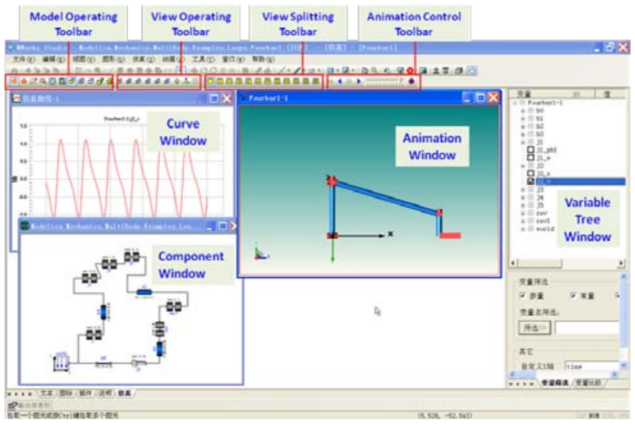


Figure 3.4 Animation Interfaces of MWorks

4 Examples

ACIS and HOOPS-based animation post-processor of MWorks has been successfully applied to simulation of MultiBody system based on Modelica.

4.1 Examples from Standard MultiBody Library

Take model Modelica.Mechanics.MultiBody.Examples.Loops.EngineV6 as an example. The results are shown in Figure 4.1 after the model is compiled, analyzed and solved. Figure 4.2 shows the results of another example of Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.fullRobot. In post-processor window, the left is its axonometric view in shadow mode, the upper-right is its front view in hidden mode and the bottom-right is its default view.

4.2 Application in Aircraft Field

Cooperating with Commercial Aircraft Corporation of China, Ltd., MWorks accomplished the simulation of aircraft landing gear under various working conditions. (See Figure 4.3)

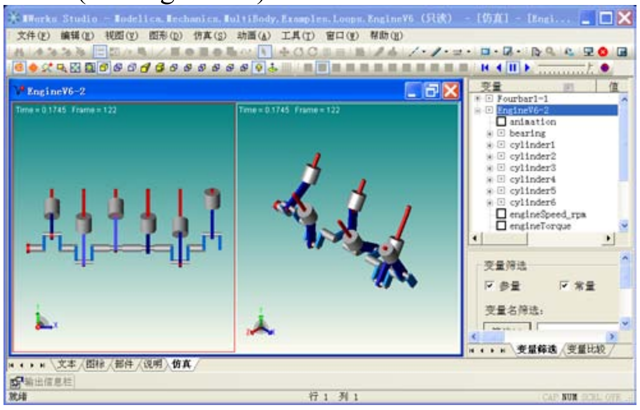


Figure 4.1 Animation of Example EngineV6

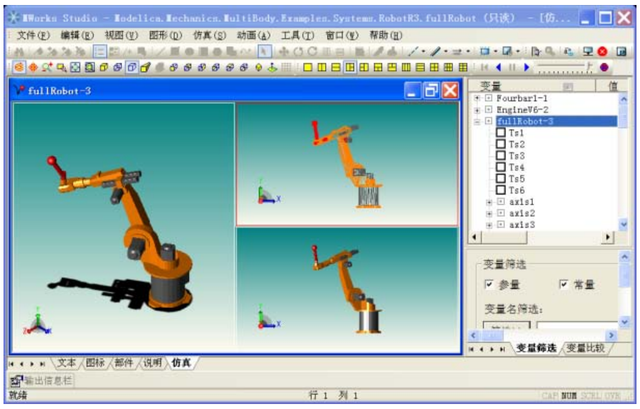


Figure 4.2 Animation of Example fullRobot

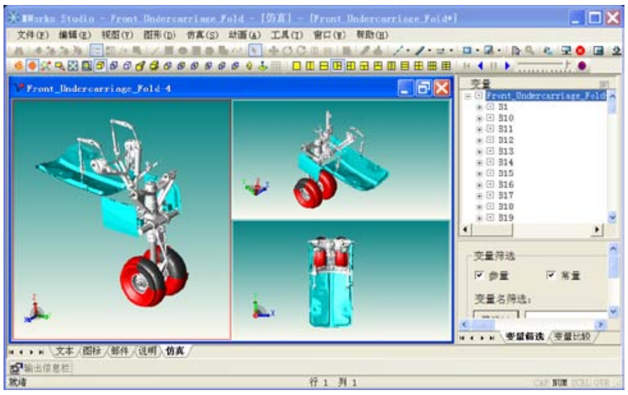


Figure 4.3 Animation of Aircraft Landing Gear

5 Conclusions

Based on ACIS and HOOPS, MWorks platform implements an animation post-processor for multibody systems. It has the advantages of convenient human-computer interaction, real-time geometric rendering, high fidelity animation effects, powerful model data management and good expandability, and has been successfully applied to some practical projects. The further work of animation post-processor of MWorks is to support flexible multibody animation, which has two tasks: (1) providing interface for common finite element software such as ANSYS and ABACUS; (2) supporting the animation of flexible body in flexible multibody library in Modelica.

Acknowledgments

The paper was supported by National Nature Science Foundation of China (No.60874064, No.60704019), National Science & Technology Major Project of China (No.2009ZX040001-015).

References

[1] ACIS online help. Spatial Technology Inc. http://www.spatial.com.

[2] HOOPS 3D Application Framework. HOOPS online help. Tech Soft American Inc. http://www.hoops3d.com.

[3] Zan Wang, Chao Xu, Xiang Xue. The Visualization Interaction Between ACIS and HOOPS. Group Technology & Production Modernization 2006, 1(23): 49 – 51.

[4] Hong-Wei Dong, Ru-Rong Zhou, Lai-Shui zhou. Developing 3D Application Software Based on ACIS. COMPUTER AIDED ENGINEERING 2002, 12 (4): 53 – 58.

[5] Ding Jianwan, Chen Liping, Zhou Fanli. A Component-based Debugging Approach for Detecting Structural Inconsistencies in Declarative Equation based Models. Journal of Computer Science & Technology, 2006, 21(3): 450-458

[6] FAN-LI Zhou, LI-PING Chen, YI-ZHONG Wu, JIAN-WAN Ding, JIAN-JUN Zhao, YUN-QING Zhang. MWorks: a Modern IDE for Modeling and Simulation of Multidomain Physical Systems Based on Modelica. Modelica 2006, September 4th – 5th: 725-732.

[7] Bo-Xing Wang, Bo Wang, Yun-Qing Zhang. Model Management in Complicated Mechanical System Simulation Platform. Journal of Computer-Aided Design & Computer Graphics, 2004, 16(4): 820 – 825.