Template-Based Traditional Building Component Modelling

Jae Woo Kim*, Kyung-Kyu Kang*, Ji Hyung Lee*
*Electronics and Telecommunications Research Institute, Korea
jae_kim@etri.re.kr, kangk2@etri.re.kr, ijihyung@etri.re.kr

***Abstract*— Creating 3D object models is a crucial in many areas such as computer graphics, virtual reality, animations, and computer aided design(CAD). In this paper, we discuss our on-going research on creating traditional building component modelling using component templates. We developed a prototype system that can create building component templates by editing fundamental 3D object primitives using the open source CAD software FreeCAD. Once a building component template is created, it will be segmented and anlalyzed to select the unit shapes that are commonly and frequently used for a variety of component templates, and they will be registered back into the primitive database so that they can be used in creating other templates in the future. Our system provides easy-to-use editing tool that a user can create the shape of the building component and then the system automatically generate a set of parameters necessary to describe the shape. Our experience showed that users can easily create the component templates they desire to make in a few minutes.**

***Keywords*—Primitive, Template, 3D, Component, Modeling,**

I. INTRODUCTION

With the recent advances in digital technologies such as computer graphics, multimedia, virtual reality, and internet technology, the demand for digital heritage preservation and dissemination is rapidly growing. Current research on digital heritage preservation for traditional buildings focused on representing overall shape of the heritage buildings in digital form ignoring modeling of detailed building components. However, from the viewpoint of digital preservation of heritage buildings, modelling of each building component that constitutes a heritage building and how the components are connected are also significant and valuable.

More recently, a data-driven approach to create 3D geometric models in which users can construct 3D models by searching a 3D model database and select and recombining parts of the models. This approach provides easy-to-use modelling capability while maintaining the quality of the models that comes from existing 3D models in the database [1],[2],[3],[4],[5],[6].

In this paper, we developed an easy-to-use template-based building component modeling tool that creates 3D building component models for heritage buildings by interactively editing 3D object primitives provided by the software using graphical user interface. Users can create building component model templates by arranging and compositing model primitives in 3D space and generate component models by modifying the parameters of the template that define the shape of the models. The software consists of model primitive creation tool, model template editing tool, and databases for model templates and model primitives.

Model template editing tool provides a graphical user interface in which users can select object primitives that include the most basic 3D objects such as box, cylinder, sphere, cone, and torus. Users translate, rotate, and resize the selected primitives and arrange them in the 3D space to define the shape of a building component as they desire. Every actions users take during the editing process will be recorded as a script statement and saved into a script file. The software will analyze the script file and conduct geometric analysis to understand the structure of the shape of the building component the user has defined. Based on the geometric information the system will automatically define the parameters that can specify the shape of the building component template. The template data consisting of geometric information and parametric information will be finally saved and registered into the template database. Once a template is created, it will be segmented into unit shapes and commonly used shape parts are selected and registered as a

new primitives into to primitive database for the future use.

Our software provides useful capabilities for creating building component models. Non-experts in 3D modeling can build the building component models by simply defining the shape of the component model templates by selecting and arranging 3D object primitives provided by the system.

II. Building Component Template Generation

We implemented our system on top of the capabilities provided by FreeCAD. FreeCAD is an open source CAD software that provides a variety of CAD functionalities such as sketching, part design, part assembly, rendering, and simulation.

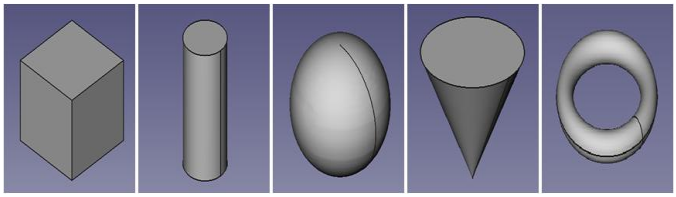

Figure 1. 3D object primitives that FreeCAD provides

A. Databases for 3D object primitives and building component model templates

The proposed software system provides two different types of databases. One is 3D object primitive database and the other is building component template database. The 3D object primitive database stores a set of fundamental 3D object primitives such as box, cylinder, sphere, cone, and torus as shown in figure 1. Those primitives are used as fundamental building blocks in constructing building component model template generation. In other words, a user can search the database, select several primitives, and composite the shape of the building component model template as she desires by arranging the primitives in 3D virtual space. Therefore, various complicated shapes can be built by compositing those primitives in 3D space.

Once a building component model template is created, it will be registered into the building component model template database. The template database therefore stores a set of component model templates that consists of geometric information to define the shape of the template and a set of parameters that can modify the shape as the user desire. The shapes in the template database are segmented into the fundamental parts and they are further analyzed to extract commonly used parts. The extracted commonly and frequently used parts are stored back into the primitive database for the future use. Figure 2 provides the overview of our building component model template generation system.

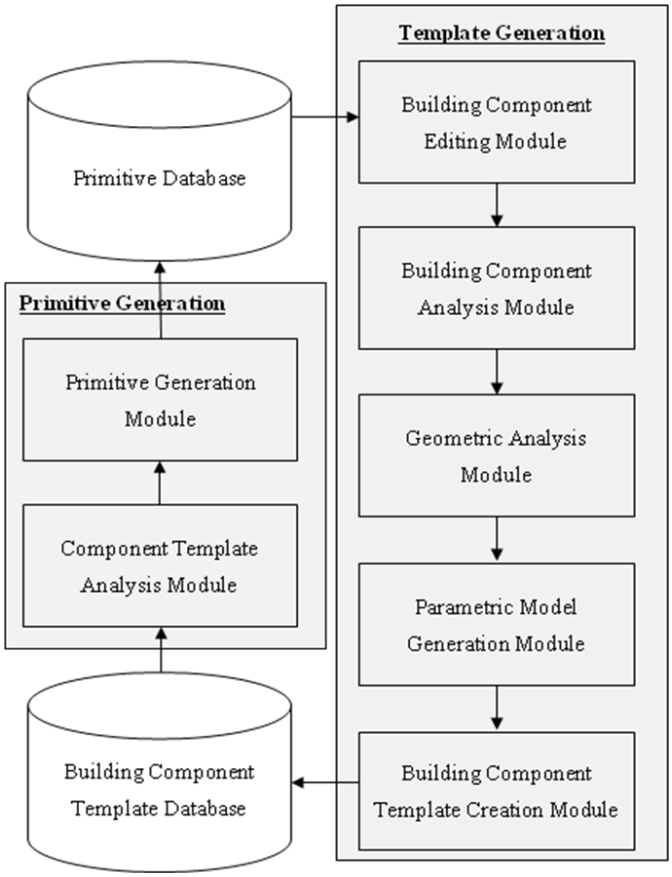

Figure 2. Building component template generation system overview

B. Building Component Model Template Generation

Our system take advantage of the capabilities the open source software FreeCAD provides [7]. A user can search and select primitives from the primitive database and edit them by translating, rotating, resizing and then aligning them in 3D space using Part Workbench the FreeCAD provides. All the user's behaviour are recorded using python script language and saved as a macro file.

The script file contains a series of python script statements that describe commands related to 3D model primitive definitions, geometric

transformations, and user interface control. Our system parses each statement line by line and divide it into tokens. Semantic analysis of the system firstly ignores the statements that is irrelevant to the geometric descriptions of the object primitives. Then it translates a set of tokens of each statement that is relevant to the geometric descriptions of the primitives into a data structure whose data fields include statement type, document id., operator name, object id., and parameters for the operator.

Statement in the python script are commands that edit the primitives in 3D. Figure 3 shows an example of a FreeCAD macro file.

```
# Macro Begin: I:\Code\FreeCAD\Scripts\pillar_example.FCMacro +++++
import FreeCAD
import Part

App.ActiveDocument.addObject("Part::Box","Box")
App.ActiveDocument.recompute()
#Gui.SendMsgToActiveView("ViewFit")
FreeCAD.getDocument("Unnamed").getObject("Box").Height = 9.00

FreeCAD.getDocument("Unnamed").getObject("Box001").Length = 9.00
App.ActiveDocument.addObject("Part::Box","Box")
App.ActiveDocument.recompute()
#Gui.SendMsgToActiveView("ViewFit")
FreeCAD.getDocument("Unnamed").getObject("Box002").Height = 9.00
FreeCAD.getDocument("Unnamed").getObject("Box001").Placement =
App.Placement(App.Vector(0,0,1),App.Rotation(0,0,0,1))
FreeCAD.getDocument("Unnamed").getObject("Cylinder").Placement =
App.Placement(App.Vector(3,3,3),App.Rotation(0,0,0,1))
FreeCAD.getDocument("Unnamed").getObject("Cylinder").Radius = 1.00
#Gui.activeDocument().activeView().viewTop()
#Gui.activeDocument().activeView().viewRight()
#Gui.SendMsgToActiveView("ViewFit")
#Gui.activeDocument().activeView().viewAxometric()
App.ActiveDocument.addObject("Part::Cylinder","Cylinder")
App.ActiveDocument.recompute()
#Gui.SendMsgToActiveView("ViewFit")
App.ActiveDocument.addObject("Part::Box","Box")
App.ActiveDocument.recompute()
#Gui.SendMsgToActiveView("ViewFit")
App.getDocument("Unnamed").removeObject("Cylinder001")
FreeCAD.getDocument("Unnamed").getObject("Box003").Height = 9.00
```

Figure 3. An example of FreeCAD macro statements

Building component analysis module firstly divides each statement into tokens and then translate a series of tokens of each statement into a data structure that represents the semantics of it. The module finally set the assembly tree that describes the whole component that consists of the primitives. Figure 4 shows an example of the results obtained by building component analysis.

```
statement 159 :
FreeCAD.getDocument("Unnamed").getObject("Box003").Placement
= App.Placement(App.Vector(1,1.5,13),App.Rotation(0,0,0,1))

FreeCAD Statement Lefthand:

[['FreeCAD'], [['getDocument'], ['Unnamed']], [['getObject'],
 ['Box003']], ['Placement']]

FreeCAD Statement Righthand:

[['App'], [['Placement'], ['App'], ['Vector'], ['1']], ['1.5'],
 [['13'], ['App'], ['Rotation'], ['0']], ['0'], ['0'], ['1']]

Semantics of the statement:

ID Number = 141
Stat Type = FreeCAD
Document  = Unnamed
Operator  = ChangePlacement
Object    = Box003
Placement = [1.0, 1.5, 13.0]
```

Figure 4. An example of results obtained by building component analysis

Geometric analysis module reads the assembly tree and analyze the relative relationships between primitives to detect the minimum number of feature points that can describe the shape of the building component. A set of parameters are determined based on those feature points and a new template is created.

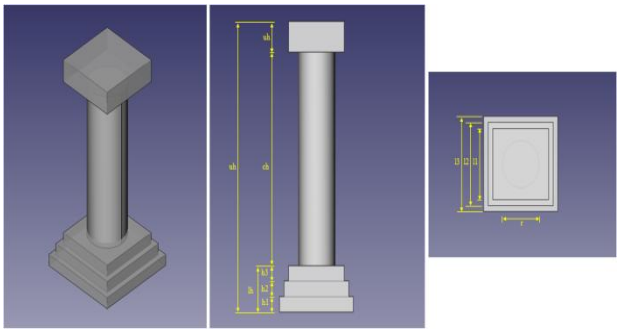

Figure 5. Building component template generation system overview

The system then analyze the script file to extract the geometric information of the primitives used in the editing process resulting in data structure that describes detailed geometric information about the primitives. The system finally generate a new set of parameters that can define the shape of the object the user constructed and they will be stored into the template database. Figure 5 shows an example of the shape of the building component template a user created and the parameters that define the shape generated by the system.

C. 3D Object Primitives Generation

The shapes of templates in the database are segmented into fundamental unit objects using 3D model segmentation algorithm [8]. The template database also stores those fundamental objects and clusters them into categories according to their shapes using similarity measures defined in [9],[10]. Here, the objects that has the same structure or shape belong to the same category based on the similarity measures. The system then count the number of objects in the same category and if it is larger than a threshold value, that is set to ten in our system, it will be regarded as a commonly or frequently used primitive. Those commonly used primitives will be stored into the primitive database so that they can be used in the future modelling process.

III. CONCLUSIONS

We developed a building component template generation system that allows a user to interactively edit the shape of the building component as she desires and automatically generates a set of parameters necessary to define and modify the shape of the component.

The benefits of our system is as follows. Firstly, it provides an easy-to-use graphical user interface that enables effective component shape design intuitively using the 3D object primitives. This capability reduces the time and effort that is required by the current 3D modelling software systems. Secondly, it provides the automated parameter detection and generation capability that is useful in designing 3D parts in CAD applications. Thirdly, the system provides means to maintain the component templates using template database. Users therefore can reuse the templates in developing 3D objects for any other applications. Lastly, our system analyze the templates in the database and select commonly and frequently used primitives to add them into the primitive database. The primitive database therefore grows as the users create more templates and the system can provide more variety of primitives that can make it easier to design the templates.

ACKNOWLEDGMENT

This research is supported by Ministry of Culture, Sports and Tourism(MCST) and Korea Creative Content Agency(KOCCA) in the Culture Technology(CT) Research & Development Program (Project ID: R2013040060).

REFERENCES

[1] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin, "Modeling by example," ACM Transactions on Graphics 23, 3.

[2] J. Lee, and T. Funkhouser, "Sketch-based search and composition of 3D models," In Proc. Eurographics Workshop on Sketch-Based Interfaces and Modeling.

[3] S. Chaudhuri, and V. Koltun, "Data-driven suggestions for creativity support in 3D modeling," ACM Transactions on Graphics 29, 6.

[4] A. Jain, T. Thormahlen, T. Ritschel, and H.-P. Seidel, "Exploring shape variations by 3D-model decomposition and part-based recombination," Computer Graphics Forum 31, 2.

[5] S. Chaudhuri, E. Kalogerakis, L. Guibas, and V. Koltun, "Probabilistic reasoning for assembly-based 3D modeling," ACM Transactions on Graphics 30, 4.

[6] K. Xu, H. Zheng, D. Cohen-Or, L. Liu, and Y. Xiong, "Photo-inspired model-driven 3D object modeling," ACM Transactions on Graphics 30, 4.

[7] http://www.freecadweb.org/

[8] E. Kalogerakis, A. Hertzmann, and K. Singh, "Learning 3D mesh segmentation and labeling," ACM Transactions on Graphics 29, 4.

[9] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The princeton shape benchmark," In SMI'04, 167-178.

[10] J.W. Tangelder, and R. C. Veltkamp, "A survey of content based 3D shape retrieval methods," Multimedia Tools Appl. 39, 3, 441-471.

Jae Woo Kim is a senior researcher at Computer Graphics Research Section, Visual Content Research Department, Creative Content Research Laboratory, Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. He received his D.Sc. in Computer Science from the George Washington University and MS. in Computer Science and Bs. in Physics from Hankuk University of Foreign Studies. His research interest includes geometric modeling, computer animation, and visualization.

Kyung-Kyu Kang is a researcher at Computer Graphics Research Section, Visual Content Research Department, Creative Content Research Laboratory, Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. He received his Ph.D, MS, and BS in Media Engineering at Soongsil University in 2013, 2006, 2004. His interests include real-time rendering algorithms and physically-based simulation.

Ji Hyung Lee is a principal researcher at Computer Graphics Research Section, Visual Content Research Department, Creative Content Research Laboratory, Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. He received his Ph.D in Computer Eegineering at Chungnam National University and MS in Computer Science at Korea University in 2011, 1996. His interest includes computer graphics and digital imaging.