International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2163
Volume 1 Issue 7 (August 2014) http://ijirae.com

Procedural Function based modelling of three dimensional objects Modelling three dimensional primitives using node-based architecture

S.Lakshmi Piriya,
Department of Computer Science,
Research Scholar (Bharathiar University),
Coimbatore, India.

Dr.S.K.Mahendran
Director, SVS Institute of Computer Applications
Anna University, India

Abstract*— A new function-based procedural representation to model heterogeneous objects containing internal volumetric structures with smaller magnitude compared to the overall size of the object is developed. This representation provides compact, precise, and arbitrarily parameterized models of coherent microstructures, which can undergo blending, deformations, and other geometric operations, and can be directly rendered and fabricated without generating any auxiliary representations such as polygonal meshes and voxel arrays. Since objects are defined as mathematical functions as opposed to a list of points, models are resolution independent and can be polygonised at any desired level of detail. Functional Representation (FRep) modelling functionality has been integrated into Houdini and its node-based environment. The library is developed in C++ using the Houdini Development Kit (HDK) and comes as a set of custom nodes.

Keywords: microstructure, function representation, node-based environment.

I. INTRODUCTION

In the visual effects and games industry, the predominant modeling techniques are using polygonal meshes. While these offer various benefits and are well integrated into current pipelines, they only represent surfaces and have a given amount of detail. As a consequence, information about an object internal structure cannot be stored and modelling of heterogeneous real-world objects is difficult. In other fields like CAD and 3D printing, the tradeoffs cannot be circumvented without efforts and alternative representations are needed. One alternative is to use “Function representation in geometric modelling” as suggested by Pasko et al. (1995). The so-called FRep allows the definition of objects, operations and relations as a mathematical function as opposed to point data and meshes. This document describes the design and implementation of a custom plug-in for Houdini that was written in C++ using the Houdini Development Kit (HDK). Houdini is a high-end 3D animation package developed by Side Effects Software and used heavily in the VFX (Visual Effects) industry. Due to its node-based architecture, it is very suitable for procedural modelling. Moreover, many of the concepts in the FRep API, like for example the tree and node structure, translate and integrate very well into the software.

II. PRIOR WORK

A. Overview of Representations

The most dominant way of representing objects in modern computer graphics are boundary representation models, also known as BRep [2]. Such models do not store any information about object’s inner properties. Figure 1 shows a triangle mesh, a popular type of a boundary representation model. In film and games, most efforts are spent on creating visually pleasing results rather than accurate models of reality. Therefore, boundary representation are often sufficient and even preferred because they offer a rich set of operations and are comparatively easy to render. For other purposes, volumetric representations as in Figure 2 allow us to store the object’s internal structure rather than only limited surface information. This is especially important when modeling real-life heterogeneous objects. There are a number of volumetric representations including voxel representation, implicit surfaces and FReps.

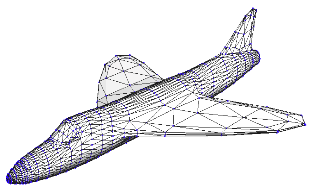

Figure 1: Example of a boundary representation: triangle mesh model [L.14].

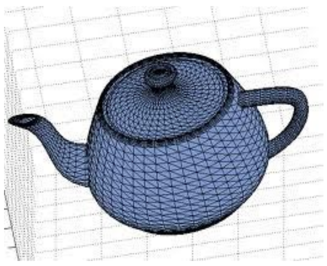

Figure 2: Example of a boundary representation: triangle mesh model [L.14].

The Functional Representation(FRep)

The Functional Representation (FRep) combines different models such as algebraic surfaces; skeleton based “implicit” surfaces, CSG (Constructive Solid Geometry), sweeps, volumetric objects, parametric models and procedural models. The representation defines a geometric object by a single real continuous function of point coordinates as $F(X) \geq 0$ (Pasko 2011) [2]. This also lets us represent models independent from resolution. A constructive tree defines functions and provides a visual overview of operations and parameters. Leaf nodes are primitives like a box, sphere, torus, etc. Non-leaf nodes contain operations and relations. This functionality was implemented and made accessible via a FRep API.

One particular class of the FRep operations is set-theoretic ones defined by Real functions [L.25]. An object resulting from the set-theoretic operations has the defining function expressed as follows:

$f3 = f_1 \vee f_2$ for the union

$f3 = f_1 \wedge f_2$ for the intersection

$f3 = f_1 \setminus f_2$ for the subtraction

Where f_1 and f_2 are defining functions of initial objects, f3 is the final object obtained and the symbols v,v, / are signs of Real functions.

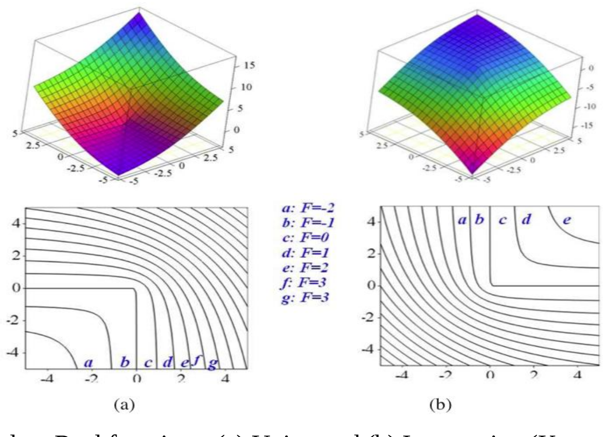

Figure 3: Set-theoretic operations based on Real functions: (a) Union and (b) Intersection (Kravtsov 2011) [2]

PROCEDURAL FUNCTION BASED MODELLING IN HOUDINI

Modelling using FRep nodes is very similar to the traditional approach of procedural modeling in Houdini. FRep nodes can be classified as primitives and operations. In order to keep the workflow as natural as possible without forcing users to adapt to new nodes, FRep Houdini’s built in primitives Box, Sphere, Torus, Cone and Tube (called Cylinder in the FRep API) can be used.[L.27] Making use of Houdini’s native primitive nodes also enables us to reuse guide geometry and handles.

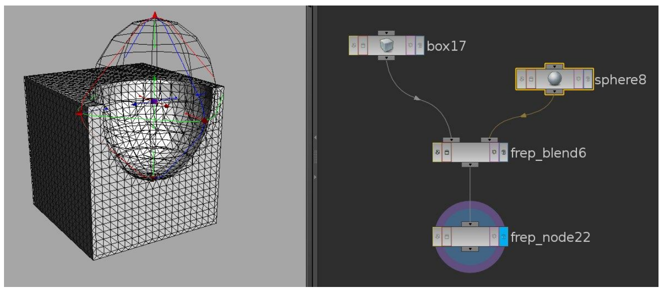

Figure 4: Using built-in primitive types as guide geometry

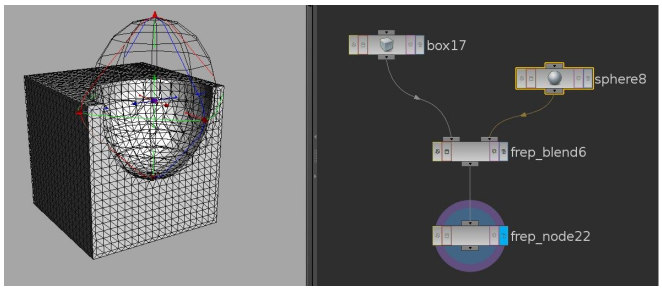

Figure 5: Transform node for scale, translation and rotation

TREE TRAVERSAL

Gathering the necessary information to build the FRep tree in Houdini architecture requires every node to be independent, so all the attributes need to be passed and kept up to date.[L.28] In Houdini, this is slightly easier to integrate and one node can be used to gather all its input nodes recursively and build the tree from that. This node at the end of the tree (in Houdini typically at the bottom) is represented by the FRep_Node class. The traversal of the nodes is controlled by one single, recursive function called retrieve Children. Not only is this function responsible for traversing all child nodes, it also creates the corresponding FRep entities and nodes as well as fills them with the parameters provided by the user in Houdini. The function is run with the (only) child of the FRep_Node as input and returns a full FRep tree consisting of FRep nodes and their children. The algorithm can be structured in five phases:

1. Variables like factory and return Node are initialized. Also, further flags to indicate the type of the current node are defined.

2. A new frepNode instance is created. This is achieved either by casting the current node, if it is an actual FRep node, or by manually creating it in case the current node is one of Houdini's built-in primitives (Sphere, Box, Torus, Cone, etc.). Additionally, a new FRep entity of the corresponding type is created. The node holding that entity is also marked to be returned.

3. The FRep API offer a transform entity which can be appended to a node to perform operations. In case we are using an FRep node that can be transformed or even Houdini's own transform SOP, create the necessary FRep entity. Then, create the corresponding node and add it as child of the original node. In this manner, an "invisible" FRep transform is added right after the node it transforms.

4. In the last step, the algorithms determines what to return. If the current node is an FRep object (either by being one of the FRep nodes or one of Houdini's primitives representing one), fill the entity and run this whole procedure for every child of this node. Alternatively, if this is just an ordinary node that should be ignored by these systems, run retrieve Children on its input (child). If the node is irrelevant to the FRep system and has no inputs, return NULL. [L.27]

Algorithm

```
 **Function: retrieveChildren (thisNode)**
// Phase 1: Initialization
factory ← Create factory
returnNode ←Node to be returned (initially NULL)
isFRepNode ← Is the node one of our own FRep nodes?
```

```
isHoudiniGeoNode ← Is Houdini geometry node like Sphere, Box, Torus, etc.?
isTransformNode ← Is Houdini transform node?
// Phase 2: Create FRep node and entities
if isFRepNode or isHoudiniGeoNode then
if isFRepNode then
frepNode ← cast thisNode as FRep_Node
else if isHoudiniGeoNode? then
frepNode ← create FRep node for Houdini native primitive
end
newEntity ← create entity of frepNode.getTypeId() via factory
newFrepNode← create node based on newEntity
returnNode ← newFrepNode
end
// Phase 3: Add transform node
if isFRepNode or isTransformNode then
transformEntity ← create entity with the transforms of thisNode
transformNode ← create node based on transformEntity
transformNode.addChild (newFrepNode)
returnNode ← transformNode
end
// Phase 4: Fill parameters and run on child nodes (if any)
if isFRepNode or isHoudiniGeoNode then
frepNode.fillParameters ()
for each child in inputNodes do
childNode ← retrieveChildren (child)
if Child is FRep node then
newFrepNode.addChild (childNode)
end
return
returnNode
else if thisNode has input then
return retrieveChildren (input)
else
return NULL
end
```

Experimental Results

Like traditional polygon modelling, FRep modelling offer the same vastness of possibilities. Virtually anything can be created, any attribute of an object can be controlled and changed over time. To demonstrate some of the modelling capabilities, a real-world object (Figure 6) and a simple character (Figure 7) have been created. As shown in the example of the screw, such an object can be created with very few nodes.

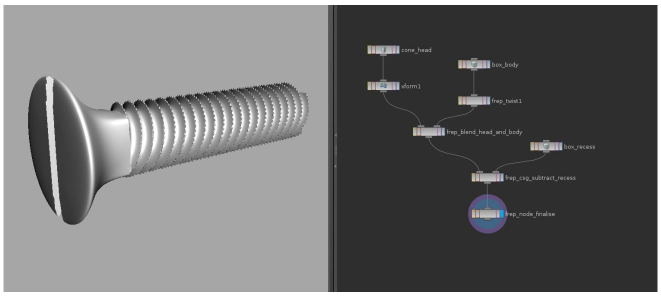

Figure 6: Screw model with its corresponding node network and optimization for sharp features

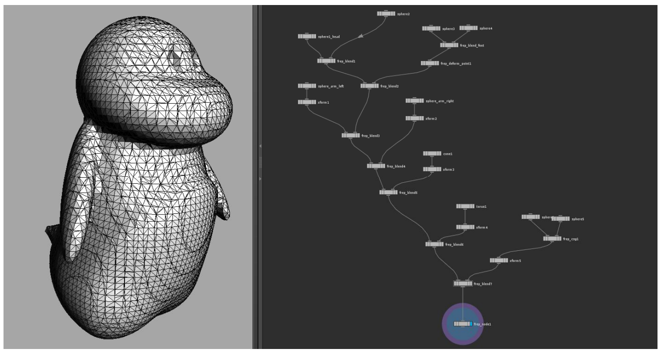

Figure 7: Penguin FRep model

CONCLUSION

Developing a project on an external library, while at the same time tackling a complex Software Development Kit like the Houdini Development Kit was certainly an ambitious goal. In retrospect, the objective of bringing FRep modelling capabilities to Houdini while respecting existing guidelines and User Interface paradigms was achieved. Particularly the degree of integration into the Houdini environment is very promising for future projects and applications. The FRep API offer a vast set of features. The technology can have applications in industries like film and games, especially as computers get faster and more accurate representations of the real-world are desired. Furthermore, being able to blend and morph objects regardless of topology is challenging using current methods. Since FReps are resolution-independent, they are ideal for use in digital manufacturing and 3D printing. Additional nodes could be developed to allow export from Houdini to various file formats like Stereolithography.

ACKNOWLEDGMENT

The authors would like to thank Dr.Alexander Pasko for the FRep based model. We also thank Denis Kravtsov for providing us with the idea and base for this project through his Hybrid modelling of time-variant heterogeneous objects.

REFERENCES

[1]. Alan Watt, 3D Computer Graphics, 3rd ed. ISBN 0-20-139855-9.
[2]. Alexander Pasko, Oleg Fryazinov, Turlif Vilbrandt, Pierre-Alain Fayolle, Valery Adzhiev Bournemouth University, UK Digital Materialization Group, Japan, and Uformia AS, Norway University of Aizu, Japan, Procedural Function-based Modelling of Volumetric Microstructures.
[3]. Angel, Interactive Computer Graphics. ISBN 0-201-85571-2.
[4]. Y. Chen, 3d texture mapping for rapid manufacturing,Computer-Aided Design & Applications 4 (6) (2008) 761–771.

[5]. C. K. Chua, W. Y. Yeong, K. F. Leong, Rapid prototyping in tissue engineering: a state-of-the-art report, in: Proc. 2nd Int. Conf. on Advanced Research in Virtual and Rapid Prototyping, 2005, pp. 19–27.
[6]. Foley, van Dam, Feiner, and Hughes. Computer Graphics: Principles and Practice, 3rd edition in C. ISBN 02013985590.
[7]. Fredo Durand, A short introduction to computer graphics, MIT Laboratory for computer science, (2005) 1-10.
[8]. V. Gervasi, A. Schneider, J. Rocholl, Geometry and procedure for benchmarking SFF and hybrid fabrication process resolution, Rapid Prototyping Journal number 11(1)(2005)4–8.
[9]. Hearn and Baker, Computer Graphics, C Version, 2nd ed. ISBN 0-13-530924-7.
[10]. X. Y. Kou, S. T. Tan, Heterogeneous object design: an integrated cax perspective, in: Heterogeneous objects modelling and applications: collection of papers on foundations and practice, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 42–59.
[11]. M. W. Naing, C. K. Chua, K. F. Leong, Y. Wang, Fabrication of customised scaffolds using computer-aided design and rapid prototyping techniques, Rapid Prototyping Journal 11 (4) (2005) 249–259.
[12]. W. Sun, B. Starly, J. Nam, A. Darling, Bio-cad modeling and its applications in computer-aided tissue engineering, Computer-Aided Design 37 (11) (2005) 1097–1114.
[13]. H. Wang, Y. Chen, D. W. Rosen, A hybrid geometric modeling method for large scale conformal cellular structures, ASME Conference Proceedings 2005 (47403) (2005) 421–427.
[14]. Woo, Neider, and Davis. OpenGL Programming Guide, 4th edition. ISBN 03211734812.

Reference LINKS

[L.1] dl.acm.org/citation.cfm?id=2027562
[L.2] doc.gold.ac.uk/uk-korea-geomod2011/Slides/10-Alexander-Pasko.pdf
[L.3] en.wikipedia.org/wiki/Procedural_modeling
[L.4] eprints.bournemouth.ac.uk/18563
[L.5] eprints.bournemouth.ac.uk/9808/1/TR-NCCA-2009-02.pdf
[L.6] garyhodgson.com/reprap/2012/.../screenshot_2012-06-24_200220/
[L.7] http://en.wikipedia.org/wiki/File:Utah_teapot_simple_2.png
[L.8] https://engineering.purdue.edu/
[L.9] https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=44124
[L.10] jap.aip.org/resource/1/japiau/v97/i10/p104104_s1
[L.11] libra.msra.cn/.../bounded-blending-for-function-based-shape-modeling
[L.12] ncca.bournemouth.ac.uk/research/?sub=47
[L.13] ntrs.nasa.gov/archive/nasa/.../20050196627_2005195159.pdf
[L.14] people.csail.mit.edu/fredo/Depiction/1_Introduction/reviewGraphics.pdf
[L.15] topaz.ethz.ch/function/web-het-secured/pdfs/Kulkarni-03.pdf
[L.16] ucd-ie.academia.edu
[L.17] web.cs.wpi.edu
[L.18] web-ext.u-aizu.ac.jp/official/researchact/annual-review/2010/.../cgl.html
[L.19] www.hyperfun.org
[L.20] www.informatik.uni-trier.de/~ley/db/indices/
[L.21] www.linkinghub.elsevier.com/retrieve/pii/S1524070311000087
[L.22] www.manufacturingthefuture.co.uk/VRP2011_CellularStructure_Paper_26_May_2011.pdf
[L.23] www.nae.edu/Publications/Bridge/57865/58052.aspx
[L.24] www.physics.udel.edu/~bnikolic/.../nanoscale_device_modeling_datta.pdf
[L.25] www.researchgate.net/.../220632773_Procedural_function Based_modelling_of_volumetric_microstructures
[L.26] www.sciencedirect.com/science/journal/09270256
[L.27] SideFX (2011)a, Houdini development kit –architectural overview. Available from: http://www.sidefx.com/docs/hdk11.0/hdk_opbasics_architecture.html [Accessed 1 August 2011].
[L.28] SideFX (2011)b, Houdini development kit – geometry introduction. Available from: http://www.sidefx.com/docs/hdk11.0/hdk_geometry_intro.html [Accessed 1 August 2011].
[L.29] SideFX (2011)c, Extending the hou module using c++. Available from: http://www.sidefx.com/docs/houdini11.0/hom/extendingwithcpp [Accessed 3 August 2011].