Interactive Parameterised Heterogeneous 3D Modelling with Signed Distance Fields

Teemu Lindborg
NCCA, Bournemouth University, UK

Philip Gifford
NCCA, Bournemouth University, UK

Oleg Fryazinov
NCCA, Bournemouth University, UK

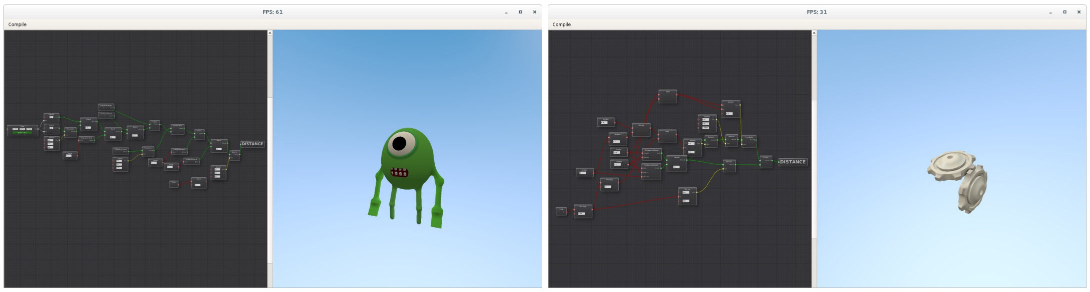

Figure 1: Examples of the models created in our system. Right: parametrised cogs, left: cartoon character.

ABSTRACT

Modern 3D modelling tools allow to create very complex scenes, however, they usually require a certain level of knowledge from users, which is not always possible for artists, especially if one needs to define internal material properties or change the parameters during the modelling process. In this paper, we present a visual, node-based approach to heterogeneous 3D modelling by using signed distance fields and material interpolation which allows for easy parametrisation of the scene with interactive rendering for a good user experience.

CCS CONCEPTS

• **Computing methodologies → Shape modeling**;

KEYWORDS

Signed distance fields, heterogeneous, parametrised, modelling

ACM Reference format:
Teemu Lindborg, Philip Gifford, and Oleg Fryazinov. 2017. Interactive Parameterised Heterogeneous 3D Modelling with Signed Distance Fields. In *Proceedings of SIGGRAPH '17 Posters, Los Angeles, CA, USA, July 30 - August 03, 2017,* 2 pages.
https://doi.org/10.1145/3102163.3102246

1 INTRODUCTION

Advances in shape modelling result in a large number of tools that allow users to define in an interactive way not only the geometry of the objects but also materials as well as the internal structure. The main problem with the majority of the modelling systems is the geometric kernel, usually based on either polygonal meshes or parametric surfaces. Being very fast with manipulation and rendering, these systems, however, lack of modelling history and access to every operation which led to the final design. As a consequence, parametrisation, i.e. an ability to change parameters for the existing object resulting in the immediate changes, is not always possible in these systems. Another limitation of the existing modelling systems is limited ability to define material inside the object, not only on the surface, which prevents true heterogeneous modelling of the objects in computer graphics systems. Alternative representations which allow avoiding several of the limitations stated above has their own disadvantages, for example, geometric kernels based on voxels require a significant amount of data to be stored on the user side, or are very slow to visualise and therefore not suitable for interactive rendering systems.

In this work, we use the representation of objects' geometry by using signed distance fields (SDF). SDF is a representation where the object is defined implicitly by a function which returns the Euclidean distance to the object. Being a continuous representation, it avoids the inaccuracy provided by explicit modelling with discrete data structures, such as polygonal meshes. This approach has proven its applicability for real-time modelling and has been used for various applications [Reiner et al. 2011]. We extend the approach presented in previous works from purely solid objects to heterogeneous objects, where the material and attributes are defined within the same model and use the node-based editor to allow a parameterised interactive 3D modelling in an artist-friendly environment. A link between tree-based model representation and direct rendering approach allows to visualise the changes in real-time and provide a good user experience which can be justified by modelling examples. As a result, we present a prototype of the interactive modelling system based on SDF to allow full parameterisation and support of materials.

SIGGRAPH '17 Posters, July 30 - August 03, 2017, Los Angeles, CA, USA

ACM ISBN 978-1-4503-5015-0/17/07.
https://doi.org/10.1145/3102163.3102246

2 METHOD OVERVIEW

2.1 Heterogeneous modelling with SDFs

Heterogeneous volumetric objects can be defined as a vector-valued function whose first component defines geometry by a signed distance field and other components define attributes, formally $\mathbf{f}(\mathbf{x}) = \langle g, \mathbf{c}\rangle, g(\mathbf{x}) = dist(\mathbf{x})$. Also, the geometric modelling system based on signed distance fields can be seen as a simplified case study of representation by a procedural tree where geometric objects (primitives) are leaves and operations are nodes. To support distance fields at all the steps of construction of such a tree, all the primitives should have distance property and the operations should either preserve a distance property or keep the distance property within a certain precision. Despite such a restriction, in the recent articles, it can be found that significant amount of primitives and operations support that [Quilez 2008; Reiner et al. 2011].

To define the material propertes, we can use either spatial enumeration or procedural texturing as a function defined per primitive. To define the material and attributes outside the objects, we adopted the approach presented in [Sanchez et al. 2015], use transfinite interpolation in-between as following. Given two primitives A and B, whose geometry and attributes are defined as $\langle g_A(\mathbf{x}), \mathbf{c}_A(\mathbf{x})\rangle$ and $\langle g_B(\mathbf{x}), \mathbf{c}_B(\mathbf{x})\rangle$ respectively, the operation $A \diamond B$ is defined as $\mathbf{f}_\mathrm{O} = \{\langle g_O, \mathbf{c}_O\rangle : g_O(\mathbf{x}) = g_A(\mathbf{x}) \diamond g_B(\mathbf{x}), \mathbf{c}_O = \frac{g_B(\mathbf{x})\mathbf{c}_A(\mathbf{x})+g_A(\mathbf{x})\mathbf{c}_B(\mathbf{x})}{g_B(\mathbf{x})+g_A(\mathbf{x})}\}$. In case of $g_A(\mathbf{x}) = 0$ and $g_B(\mathbf{x}) = 0$ simultaneously, priorities should be defined for primitives and the material for the surface of the primitive with highest priority should be used.

2.2 Parametrisation in our modelling system

The procedural tree which defines the model is represented in the interface of our modelling system by nodes in a node editor. The node editor allows to add, change the parameters of, and connect the nodes in an intuitive and visual way that allows artists to quickly and conveniently define the scene data. The node connections in the editor are colour-coded based on the data type that is being propagated, thus giving the user a quick indication of what can be connected where. The nodes are also categorised based on their function and this information is used the perform different operations when the node tree is traversed. The main features which allow for parameterised modelling are the following:
– Node editor allows and encourages users to connect the output of the node to multiple inputs of subsequent nodes. Thus by modifying a value of this single node everything attached to it would change accordingly (see figure 2).
– The part of the tree can be collapsed into a "supernode", which can be used in the graph as a normal node and be expanded back into a set of signed distance primitives if needed.
– Instancing of the nodes including "supernodes" allows copying and re-use parts of the procedural tree.

2.3 From node editor to visualisation

The graph-based representation allows for easy translation from scene graph into the function code which is useful for interactive rendering and is performed every time the user requests change to be visualised. In our implementation, we use direct rendering approach by using sphere tracing where the code is stored inside

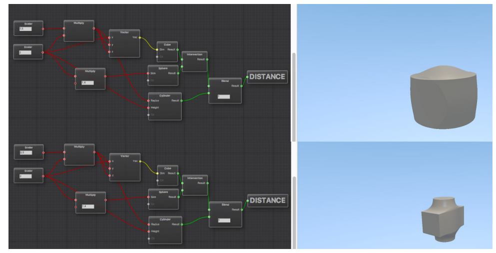

Figure 2: By changing a single value the whole nature of the object can change.

the fragment shader which allows for fast hardware-accelerated visualisation. The basic outline of the application workflow is the following: a) The user generates the scene inside the node editor using the nodes provided and connects the final result to a static "Root"-node; b) The node tree is traversed for conversion into the shader code in a top-down approach with collating appropriate fragments of the shader code as a result; the materials, transformations and other relevant inputs are updated accordingly; c) the generated shader code is compiled to a fragment shader which includes sphere tracing algorithm, shading and visual appearance of the final object. The model is rendered by applying the resulting fragment shader to a screen-sized quad.

3 RESULTS AND DISCUSSION

Our modelling system prototype was implemented in C++ with using of Qt for interface and OpenGL/GLSL for visualisation. Our modelling system allows to produce and alter complex models which would often be difficult to accomplish using an explicit modelling application. One of the examples demonstrating our application is a cog (see figure 1, right), where the model is parameterised such that the user has intuitive control on shape parameters of the object, such as a number of teeth, teeth size, inner and outer radius. As time parameter is allowed to be used in our system, the final model can be animated. Another example demonstrating material properties modelled along with geometry is a cartoon character (see figure 1, right). In our current prototype we used only colours, but more advanced material properties, such as displacement, reflectance and transparency can also be defined.

Despite the prototype has the ability to construct 3D models in a parameterised way, it has several limitations. One of the limitations is that the user is currently restricted to the set of distance-based primitives and operations. Another limitation is a lack of user input in a visual way, for example by providing an ability to move object not just by changing its transformation node, but by moving object in a visualisation window. This can be a basis for further research in the area of parametrised heterogeneous modelling systems.

REFERENCES

Inigo Quilez. 2008. Modeling with distance functions. http://iquilezles.org/www/articles/distfunctions/distfunctions.htm. (2008).
Tim Reiner, Gregor Mückl, and Carsten Dachsbacher. 2011. Interactive Modeling of Implicit Surfaces Using a Direct Visualization Approach with Signed Distance Functions. *Computers & Graphics* 35, 3 (June 2011), 596–603.
M. Sanchez, O. Fryazinov, V. Adzhiev, P. Comninos, and A. Pasko. 2015. Space-Time Transfinite Interpolation of Volumetric Material Properties. *IEEE Transactions on Visualization and Computer Graphics* 21, 2 (2015), 278–288.