Deep Nodes - creating 3D objects with node-trees

Andreas Reich*
logophoman@gmail.com
a.reich@hs-furtwangen.de
Hochschule Furtwangen University
Weilheim Teck, Deutschland

Louis Trouillier
louis.jakob.trouillier@hs-furtwangen.de
Hochschule Furtwangen University
Pforzheim, Deutschland

Matthias Mühl
matthias.muehl@hs-furtwangen.de
Hochschule Furtwangen University
Furtwangen im Schwarzwald
Deutschland

Marcel Bauer
m.bauer@hs-furtwangen.de
Hochschule Furtwangen University
Bühl, Deutschland

Lea Stegk
lea.marie.stegk@hs-furtwangen.de
Hochschule Furtwangen University
Furtwangen, Deutschland

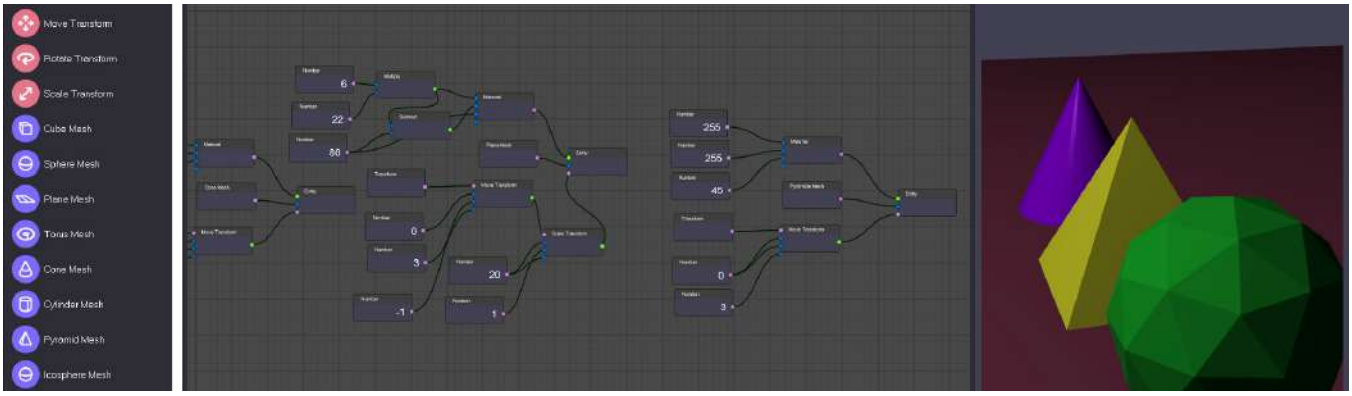

Figure 1: Excerpt form the "DeepNodes" 3D editor created during the project

ABSTRACT

We introduce *DeepNodes* - a novel node-based 3D modeling engine which we built to be used by human modelers as well as machine learning algorithms. Common 3D modeling engines are designed to be only used by humans, whereas *DeepNodes* aims to automate 3D modeling processes to help human 3D artists save time. This paper gives an overview over the editor, its techniques and its design. We built *DeepNodes* to serve as a basis for further research, especially concerning machine learning. By concatenating humanly pre-programmed functions of the editor with neural networks, complex 3D models could be re-modeled automatically.

CCS CONCEPTS

• **Human-centered computing** → *User interface design*; • **Software and its engineering** → **Software design engineering**.

Reich, MuC'21 Workshops, Hamburg, Deutschland

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

KEYWORDS

nodes, node-trees, modeling, automation of modeling, node editor, machine learning, user interface design, human-computer interaction, software engineering

ACM Reference Format:
Andreas Reich, Louis Trouillier, Matthias Mühl, Marcel Bauer, and Lea Stegk. 2021. Deep Nodes - creating 3D objects with node-trees. In *Hamburg '21: Digitaler Wandel im Fluss der Zeit, September 06–09, 2021, Hamburg*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Many modern computer programs, especially 3D modeling engines, offer users to interact with their software via so-called nodes. Nodes visually represent a sequence of computer code bundled together and are called function. Nodes can be visually connected with lines resulting in so-called node trees. They are the concatenation of multiple individual functions. Therefore, the underlying sequences of computer code of multiple functions get concatenated as well and therefore form a new program or another high-level function. Node-based 3D modeling engines allow users to create 3D meshes through combining sequences of nodes. This means that a set of functions is combined to a program that generates a desired 3D mesh.

The core idea behind *DeepNodes* was to automatize the process of 3D modeling by automatically generating node trees with reinforcement learning. Therefore, our editor is designed to serve as a basis for implementing machine learning algorithms in the future.

2 NODES AND NODE-TREES

We refer to nodes as so-called functions. Functions, often also referred to as methods, are collections of executable computer code which perform a certain task. [1] Normally a node visually represents a function in a visual editor of software. Nodes generally are atomic elements of node trees. Node trees are a collection of nodes that are usually connected with each other like a graph. They can also be utilized as single nodes themselves and are thereby describing a high-level function. They can also be referred to as programs. Connections between nodes determine the execution order of nodes and therefore also the procedure of a resulting program. Most nodes have multiple inputs and outputs whereas some nodes only have inputs or outputs. The inputs and outputs of a node represent the inputs and outputs of a corresponding function. The following figure shows a simple node tree.

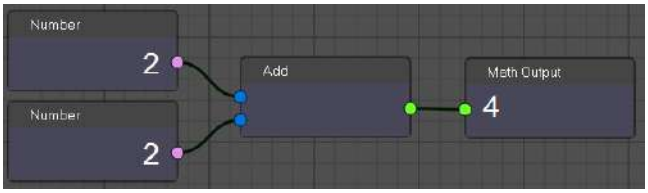

Figure 2: A simple mathematical operation with nodes in the node editor

Visual programming with node trees is increasing in popularity since, according to Blackwell, they are easier to understand than textual code. Research investigating on visual programming languages is motivated by the assumption that visual languages bear a closer relationship to the mental representation of a programmer than text based programming languages [2]. Using nodes helps programmers to visualize program routines, concepts and structures. They help them to get an overview over main tasks while still being able to focus on details [3].

3 NOVELTY OF THE APPROACH

Contrary to many other state of the art 3D modeling engines that - besides many other functions - offer 3D modeling nodes, *DeepNodes* concentrates on providing 3D modeling nodes. In comparison, the software *Blender* is an example of a node editor that uses node systems or node trees specifically to design materials of 3D objects and to connect them with textures [4]. *Maya* from Autodesk Inc. in turn uses its built-in node editor primarily for 3D character rigging and recommends using the editor less for creating materials or shaders [5].

Therefore, the *DeepNodes* 3D editor profits from its unique characteristic by focusing on modeling and creating 3D objects using node trees. Furthermore, common features like transforming 3D objects and changing the color of materials are also performed via nodes.

Independent of the current state of the node editor, which currently only allows manual modeling, our long term goal is to accelerate and automate the generation of node trees by using machine learning within our editor.

4 CREATION OF THE ENGINE

Before we decided to develop our own editor we investigated other state-of-the-art 3D-engines. Building a machine learning solution on top of the prevalent 3D-engines proved to be very complex though, especially because of a lack of documentation concerning node programming throughout all major 3D-engines. To have full control over a 3D engine environment we decided to create one of our own. We created the "`DeepNodes`" 3D editor which one can download and test here. The editor is completely node-based and capable of creating and manipulating 3D meshes via nodes. We implemented basic 3D transformation operations in order to be able to translate, rotate, and scale meshes. In order to create the software we utilized the following frameworks.

4.1 QT and PyQt

Qt is a C++ based cross-platform open-source toolkit for creating GUIs. PyQt is a binding of the toolkit for Python, implemented as a plug-in for Python. It is developed by the British company Riverbank Computing. and is available under a variety of licenses including the GNU General Public License (GPL) and a commercial license [6]. PyQt implements over 6,500 classes, functions, and methods. Our node editor heavily relies on the toolkit, especially its user interface classes and widgets.

4.2 OpenGL

OpenGL (Open Graphics Library) is a specification for a cross-platform and cross-language programming interface (API) for the development of 2D and 3D computer graphics applications. The OpenGL standard describes around 250 commands that allow the representation of complex 3D scenes in real-time [7].

4.3 Qt3D

Qt3D is a Qt module, providing support for 2D and 3D rendering. Furthermore it allows to manipulate meshes and perform transformation operations. It is written in C++ but can be accessed through the PyQt3D Module. The module is based on OpenGL [8]. We make use of the OpenGL functionalities and render pipelines in our editor through Qt3D.

4.4 Node Classes

In our engine we differentiate between 5 node classes.

- **Math operations** - numbers, constants, calculations
- **Basic shapes** - cubes, spheres, planes etc.
- **Transform operations** - translate, rotate, scale 3D objects
- **Material operations** - coloring 3D objects
- **Entities** - visualize 3D objects

5 DESIGN AND USER EXPERIENCE

To investigate the editors potential target groups, we conducted a survey [9]. Through the analysis of this survey we learned that

our participants are divided into two subgroups, each focusing on different aspects. The first subgroup are beginners in the field of 3D design and the use of node editors; the other subgroup consists of experienced 3D design experts. Based on the results of the survey, we also concluded that the graphical user interfaces as well as the usability of several frequently used 3D engines could be enhanced. This is particularly evident in the responses of many survey participants, who state that the GUI of the engine they use is partially or heavily overloaded. A further knowledge gain are the statements of some survey participants who find the documentation of the functional range of many 3D engines to be inadequate. From this we conclude that the documentation of our 3D modeling engine must be comprehensive and detailed. Therefore, we derive that the implementation of our 3D modeling engine must satisfy the demands of users, especially concerning a lightweight GUI, in order to deliver added value compared to existing, similar software.

5.1 Color spectrum

The color spectrum of the editor especially targets the subgroup of "beginners" by delivering clean, differentiable and complementary colors that are easy to understand intuitively. These colors can also easily be understood by "experts". All colors are optimized for the usage in a dark-mode setting.

We developed the following colour schemes, from which the five main colour shades (HEX colour codes 61CF60; B3FA8B; C8E1FF; 7D96EB; 6455CE) were defined. In order to construct a flexible color palette, additional tertiary colour shades were created in between, which were primarily used for contrast and complementing the previously mentioned colour shades.

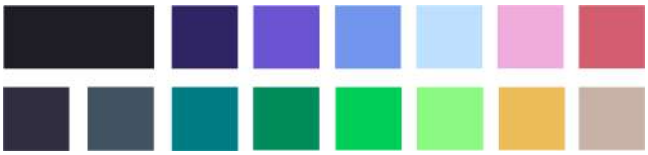

Figure 3: The color spectrum of the 3D modeling engine

5.2 Typography

The visual communication of the 3D modeling engine uses the fonts "Work Sans", designed by Wei Huang [10], and "PT Serif", designed by "ParaType" [11] for the text that appears in the Graphical User Interface. Both fonts are published on "Google Fonts". We use the sans serif font "Work Sans" for headlines, body text, description text and smaller font sizes; in sub-headlines the roman font "PT Serif" was used as a typographic contrast.

5.3 Icons and Logotype

We created several icon categories for the engine, which symbolize the functions ("nodes") of the engine. A distinction is made between core operations, elementary for creating 3D objects, object operations such as transform, zoom, move, rotate and mesh operations like "Cube-Mesh" or "Sphere-Mesh". Finally, operations such as "Number" or "Subtract" complete the scope of operations of the 3D modeling engine.

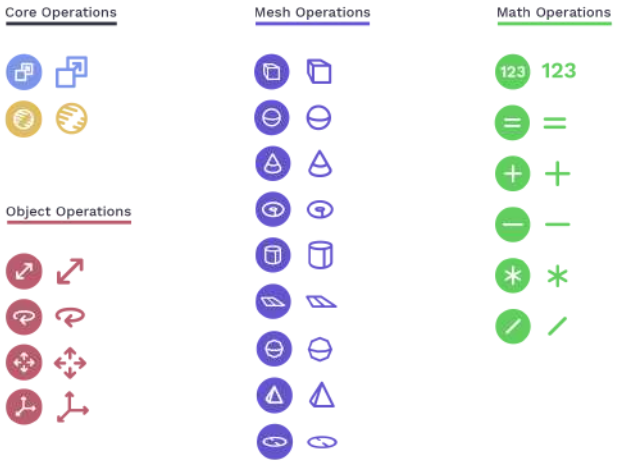

Figure 4: A summary of the icons designed for the graphical user interface

The figurative part of the logotype of our 3D modeling engine is formed by a node and a cube; the sockets, i.e. the connections to a node, are highlighted on the sides of the form and accentuated in violet color. The textual part of the logotype has a color contrast between the word parts to emphasize the deep learning focus of the project. The visual style of the logo is primarily designed for dark graphical user interfaces since the editor utilizes dark design as well.

Figure 5: The logotype.

5.4 Graphical User Interface

The Graphical User Interface (GUI) is the result of the combined thought process explained in the previous sub-chapters based on the survey results. It consists of easy-to-understand icons, which can be converted into nodes via drag-and-drop from the "Node Bar" positioned on the left interface edge (Fig. 6, red frame). The node-editing area has a grid background and the nodes can be arranged on it without any limitations (Fig. 6, green frame). Furthermore, nodes can be connected and connections between nodes can be deleted. The 3D viewport shows 3D objects created by nodes. It offers users to inspect 3D objects by rotating, panning and zooming around the object (Fig. 6, blue frame).

Users can open up new or existing files, save the current file or exit the application via the "File" menu item (Fig. 7, red frame). Basic operations like undo, redo, cut, copy and paste can be performed via shortcuts or the "Edit" menu item (Fig. 7, blue frame). The window

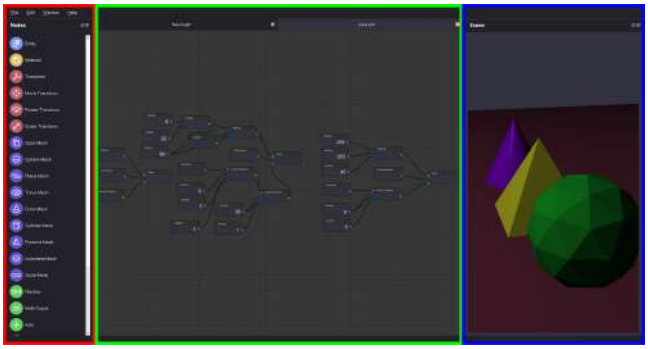

Figure 6: An excerpt from the node editor GUI.

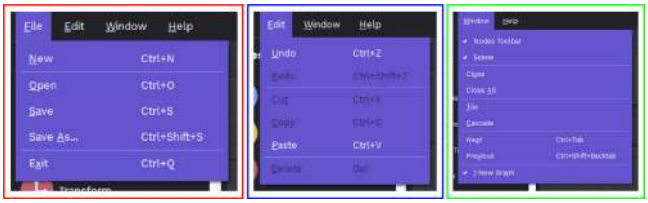

Figure 7: Menu entries with included sub entries.

can be arranged by users by hiding different interface sections like the "Node Bar" menu or the 3D viewport. The node-editing area can be tiled or cascaded to allow users to drag it around via the "Window" menu item (Fig. 7, green frame). Nodes can be dragged around and arranged at the will of users. The node-view enables users to zoom in and out to always get their desired view on node trees.

5.5 Key benefits of our human-computer interaction

Conventional software that utilizes node editors usually does not have the functionality to design complete 3D objects via node trees. Our 3D modeling engine provides the basis for implementing a neural network architecture into the modeling process of 3D objects. In this way a large number of intermediate steps that a 3D designer or engineer has to perform could be automated. This represents a step forward in the approach of conventional 3D object modeling.

Instructions on how to use our 3D modeling engine can be found in our publicly available documentation [12]. The most important aspect during the creation process of the documentation was, as already explained in 5 - "Design and User Experience", the advantage over other common software documentations by providing users with a visually comprehensible and written explanation of the basics of our software, e.g. the explanation of "nodes" and their usage. Therefore even inexperienced users can start using our node editor.

6 PERSPECTIVE ON FUTURE WORK

Currently the *DeepNodes* editor is capable of performing simple mesh generation and manipulation tasks by manually building node trees. In the future more complex operations could be added to enhance the capabilities of the engine. We created the editor in a way that one could expand it to work with machine learning algorithms and investigate if and how node-trees and the resulting 3D models could be automatically generated through neural networks. During the project, we investigated state of the art machine learning principles such as supervised, unsupervised and reinforcement learning [13]. After especially evaluating supervised learning for our project, we focused our research on reinforcement learning, since we found it to be the most promising approach.

6.1 Open tasks and questions

An open question is whether machine learning, especially reinforcement learning can be utilized to generate node trees that generate 3D meshes. Furthermore, finding an efficient algorithm that is compatible with other modern reinforcement learning solutions and that can handle the problem of reliable node-tree generation is a difficult task. Our work and editor can serve as a basis for other researchers that want to conduct research in the area of automated creation of complex programs/3D-meshes via generating node-trees with machine learning.

6.2 Prospect

Creating an efficient algorithm for node-tree generation concerning 3D models would solve a task with high practical value for the 3D industry since re-modeling meshes with nodes would simplify many VFX tasks. Moreover, such an algorithm could enhance the photogrammetry pipeline because dense point clouds could be reverted into lightweight, editable node-based models. Furthermore, findings in the area of automatic 3D modeling with node-trees and machine learning could be important for other areas of research, especially the field of neural program synthesis.

REFERENCES

[1] Qin S. Ait-Ameur Y. *Formal Methods and Software Engineering*. Springer Nature, 2019.
[2] Blackwell A. F. Metacognitive theories of visual programming. what do we think we are doing?: Proceedings ieee symposium on visual languages. 1996.
[3] Petre M. Blackwell A. F. Mental imagery in program design and visual programming. 1999.
[4] Blender.org. Introduction: Node editor. https://docs.blender.org/manual/en/2.79/editors/node_editor/introduction.html, 2017.
[5] Autodesk.Help. Autodesk knowledge network: Node editor. https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2019/ENU/Maya-Basics/files/GUID-23277302-6665-465F-8579-9BC734228F69-htm.html, 2019.
[6] Willman J. M. *Beginning PyQt: A Hands-on Approach to GUI Programming*. Apress, 2020.
[7] The Khronos Group Inc. Opengl overview. https://www.opengl.org/about/#1, 2020.
[8] The Qt Company Ltd. Qt documentation: Qt3d. https://doc.qt.io/qt-5/qt3d-index.html, 2020.
[9] Bauer M. Survey deepnodes. https://docs.google.com/forms/d/e/1FAIpQLSfVbEihCsfdno7AZDhLpDkGt2qn1HLCbalolN_jyk815nMjUA/viewanalytics, 2020.
[10] Google Inc. Work sans - google fonts. https://fonts.google.com/specimen/Work+Sans, 2020.
[11] Google Inc. Pt serif - google fonts. https://fonts.google.com/specimen/PT+Serif, 2020.
[12] Bauer M. Deepnodes 3d node editor: Dokumentation. http://deep-nodes.de/dokumentation, 2020.
[13] Lison P. An introduction to machine learning. http://folk.uio.no/plison/pdfs/talks/machinelearning.pdf, 2012.