Excel@FIT 2024
http://excel.fit.vutbr.cz

Blender Add-on For Conversion of Nodes to Vector Graphics

Filip Dráber*

Abstract
This project implements a tool that enables users of the Blender 3D modelling software to quickly and easily export Node Graphs – Blender's abstract visual programming tool – to a vector representation. This means the user does not need to take a screenshot of their work, which is considered poor practice for technical writing. The add-on parses data of the current Blender project and constructs an SVG document defining all the necessary graphical elements. The aim is to create SVG representations that allow users to re-create the graphs in their own editor. The individual images on the output are designed to hold the necessary informational value to recreate Node Graphs in a different editor, even if they are not pixel-perfect. Certain elements of various Nodes are dropped while others are enhanced to deliver all the necessary visual information which Blender would not display at once. While the add-on is still undergoing testing and feedback-driven updates, it is already used to its desired goal: exporting user-created Node Graphs for presentation purposes as part of their articles and theses.

*xdrabe09@vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Blender is a multi-purpose editor for almost anything related to 3D modelling and visualisation. Its Node Editor environments make for an excellent tool to simplify difficult concepts through visual programming (a method of using visual elements to let users create and customize applications [1] (Section II) and data flows, similar to Unreal Engine's Blueprints). They are mainly used for defining material properties, 2D composition and scene geometry, and the user-defined Node Graphs are an important element of any Blender work, be it leisure, academic, or professional.

[Motivation] When featuring Node Graphs in documentation, the user can either take a screenshot of the graph, or recreate the graph in a vector graphics editor. The former is unaesthetic, and the latter is painstaking. For writing technical reports, a user might need a simple tool that depicts their Node Graph in a vector format at the click of a button – which is exactly what this project accomplishes.

[Definition] The project aims to expose a window in Blender's user interface when a user is editing a Node Graph (Fig. 2, Step 1), where they are able to press a button to create an SVG file containing the vector representation of the currently displayed Node Graph. Multiple options are offered to the user that allow them to adjust the colors, extent of the export and other parameters of the process.

[Contribution] The ability to quickly create representations of Node Graphs has been a long-standing request by academics. The implemented tool is now being used by academics and students alike for various theses and articles in or about the Blender application, and aspires to be a mainstay of this field.

The figures present in the poster (Fig. 3 and Fig. 4 center, right) are directly exported from Blender without further adjustment.

2. Implementation

The add-on is implemented in Python, as Blender boasts architecture to integrate such scripts into its runtime. These scripts can then import the virtual `bpy` module, which encapsulates various data and types, as well as the current state of the editor and the project.

The project's core functionality is retrieving and parsing data related to the editor whence the export function is invoked, and constructing an XML tree to form a vector file of the SVG format. Its hierarchical format allows for logical grouping of certain elements together (i. e. the individual elements of a Node). Without explicit placement adjustments, SVG images are rendered using an approach where elements defined later in the file will be rendered over those defined earlier, the so-called "painter's model" [2] (Section 3.1). Ordering of the elements in the output file mirrors Blender's own rendering process, seen when inspected through tools such as RenderDoc. The poster (Fig. 1 depicts how the elements are defined: Frames (A), Wires (B), Nodes (C) and Markers (D), layered over each other.

2.1 Editor Features

To help users organize their Node Graphs, Blender has introduced various tools to clean the space up. The poster (Fig. 3) features examples of three tools used to keep a Node editor neat:

- **Node Muting**: Disabling the functionality of a Node, turning it translucent while maintaining the dataflow (A)
- **Reroute Nodes**: Nodes which allow users to rearrange their Nodes' connecting wires (called 'noodles' in Blender internally) (B)
- **Node Hiding**: Collapsing a Node whilst keeping its functionality, saving up on space in the graph (C)

2.2 Export Properties

The add-on also allows the user to configure the exporting process within Blender's user interface. These options include adjusting the color and other properties of the Nodes to visualize. This extends to their header colors, the presence of rounded corners (as Blender displays them), using an outline or the quality of more complex graphical elements of the interface. The poster showcases using two different export configurations for the same RGB Curves Node (Fig. 4), presented next to a screen capture of the Node (left): one using Blender's 'Print Friendly' preset with the export set to copy the properties of Blender's theme (center), and one with adjusted header and element colors.

3. Usage

As mentioned in the Introduction (1), the add-on has been a sought-after tool due to the importance of Node Graphs to specific Blender projects, and their abstract appearance warranting a vector representation. The usual method of simply taking a screenshot of a Node Graph (which is what Fig. 1 is) has been met with dislike due to the notorious poor quality of raster images depicting abstract shapes.

While the add-on still has not yet undergone a full release, it has been in circulation and picked up by students and academics alike. The feedback has proven invaluable in these late development stages, as exhaustive work with the Node Graph editor on the users' end has helped discover issues across multiple edge cases and application versions, while users are able to request additional functionality of the add-on that has not been intended in its conception.

4. Differences

The add-on's output does not copy the original appearance from Blender perfectly. Some of it is intentional, either adding or removing elements from the Node. The poster features an example using the RGB Curves Node (Fig. 4 and Fig. 5), where various tools around the graph are not depicted in the output, but, simultaneously, the graph showcases all four of the graph's curves at once, where Blender itself would only show the user one curve at a time.

Most other inconsistencies stem from the fact that Blender does not expose enough data to capture the exact appearance of a Node either from a data layer perspective, or as rendering calls (which has been attempted through the RenderDoc application). However, pixel measurements are not necessary as most of the information is already conveyed through the general appearance of various elements.

Acknowledgements

I would like to thank my supervisor, Ing. Tomáš Chlubna, for their help. I would also like to thank fellow students as well as professionals in the 3D modelling and animation field for helping me design and test the add-on.

References

[1] Mohammad Amin Kuhail, Shahbano Farooq, Rawad Hammad, and Mohammed Bahja. Characterizing visual programming approaches for end-user developers: A systematic review. *IEEE Access*, 9:14181–14202, 2021.

[2] Eric Willigers, Chris Lilley, David Storey, Amelia Bellamy-Royds, Dirk Schulze, and Bogdan Brinza. Scalable vector graphics (SVG) 2. Candidate recommendation, W3C, October 2018. https://www.w3.org/TR/2018/CR-SVG2-20181004/render.htmlPaintersModel.