`NetPlop`: A moderately-featured presentation editor built in NetLogo

Patrick Steinmann
Wageningen, The Netherlands
mail@patricksteinmann.com

2021-04-01

Abstract

`NetPlop` is a presentation editor built entirely in NetLogo, an agent-based modelling environment. The `NetPlop` Editor includes a variety of tools to design slide decks, and the Viewer allows these decks to be displayed to an enraptured audience. A key feature of `NetPlop` is the ability to embed agent-based models. This work has applications to climate change, sustainable development, and pandemic mitigation, as slideshows are used in all these domains.

1 Introduction

A previous contribution to this conference (Wildenhain, 2017) has shown the unexpected Turing-completeness of Microsoft PowerPoint®, a presentation editor that lets you create great presentations (Microsoft Corporation, n.d.). This implies that any given computational task can be performed entirely in PowerPoint. As a systems modeller, I found this intriguing, and considered exploring PowerPoint's universality by using it to simulate a complex system. However, that seemed like a big effort once I realized that I don't know much about PowerPoint, and virtually nothing about computer science.

The obvious consequence was to invert the problem, and create a presentation editor inside a complex systems modelling tool. For two reasons, I chose NetLogo (Wilensky, n.d.) as the environment for this. Firstly, it shares a CamelCase naming scheme with PowerPoint. Secondly, a haphazard review of the CoMSES Model Library (CoMSES Net, n.d.), a repository of agent-based models, reveals that of 860 uploaded models, 589 are tagged with the keyword "netlogo". This is incontrovertible evidence that (at least!) 68.5% of agent-based modellers use NetLogo, maximizing the impact of this work.

2 Specifications

A presentation editor must include three main features (Wikipedia, n.d.):

1. inserting and formatting text
2. inserting and formatting images
3. a slideshow system to display content

An informal peer survey revealed two further critical features:

4. free-hand drawing
5. animated slide transitions

Features which are apparently unimportant (by virtue of not having been identified by literature review or survey), yet still present in competing presentation editors, are:

6. loading and saving slide decks
7. exporting decks to a common file format, in case the computer in whatever dingy lecture hall you'll be presenting in doesn't have the requisite software installed.

3 Design

`NetPlop` consists of two separate NetLogo models: the `NetPlop` Editor, and the `NetPlop` Viewer. Splitting functionality seemed like the easiest way to handle a number of interesting ~~limitations~~ quirks of NetLogo, such as the inability to view a model full-screen. The entire functionality is available without leaving the comfort of the NetLogo Interface tab (henceforth "GUI") for the endless wastelands of the Code tab.

3.1 `NetPlop` Editor

The `NetPlop` Editor implements most of the `NetPlop` functionality through a combination of NetLogo model code, and NetLogo GUI design. Did I mention NetLogo already has a massive amount of GUI functionality? Display, console, buttons, menus, text input fields, they're all there. I'm surprised no one has done this earlier.

3.1.1 Presentation-level Editing

After creating a new presentation, its name can be specified, and it can be saved. Saving is done through a custom `.nplp` file format, the creation of which was beyond the wildest programming dreams of this humble author. A previously saved presentation can also be loaded, satisfying Specification 6. Furthermore,

an entire presentation can be exported as incrementally numbered `.png` files, satisfying Specification 7.

3.1.2 Slide-level Editing

Images and text can be added to slides with intuitive buttons, satisfying Specifications 1 and 2. On the more creative side, the currently viewed slide's background color can be specified, and then drawn upon with a variety of brush sizes. This satisfies Specification 4. All color options in `NetPlop` draw exclusively from NetLogo's extremely contrived color scheme, purposefully eschewing the option of RGB colors for extra NetLogality. For each slide, a transition animation can be specified from a rather limited selection, satisfying Specification 5.

3.2 `NetPlop` Viewer

The `NetPlop` Viewer can basically only load and cycle through existing presentation decks. However, this already satisfies Specification 3. The Viewer also displays slides in a 50% larger window than the Editor, so your audience might even be able to see them!

3.3 GUI Design

I laid out the `NetPlop` GUI to be as similar as possible to market-leading presentation editors' interfaces. To verify this, I uploaded screenshots of PowerPoint and `NetPlop` to two online image similarity measuring apps. One gave a similarity of 58.92%, the other 79.5%. Since both of those are passing grades in the Dutch school system (the former even being considered a nearly perfect score by minimalist students), I consider my job done here.

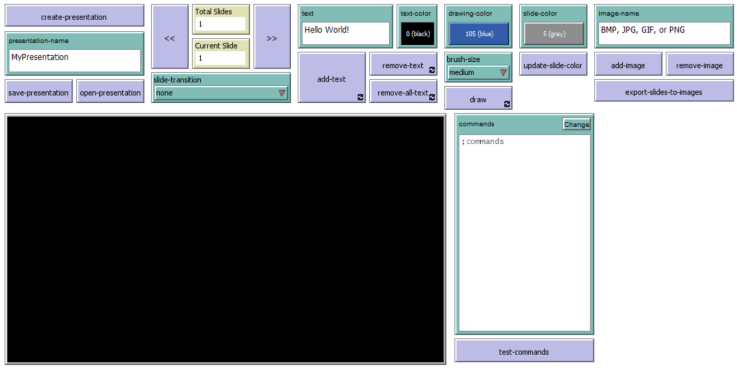

Figure 1: `NetPlop` Editor GUI. The black rectangle is the slide editing window.

Figure 2: `NetPlop` Viewer GUI. Look how much bigger the slides are displayed!

4 Richness

By virtue of having satisfied all previously defined Specifications, `NetPlop` is a complete presentation editor. However, it lacks a number of features included in competing presentation editors, such as master slides, clip art, or arrows that snap to the boundaries of whatever object is vaguely in the vicinity. Therefore, it can only be described as moderately featured.

5 Recursive Properties

5.1 Things inside Things

The *pièce de résistance* of `NetPlop` is the ability to pass NetLogo code from the GUI to the model itself. While this might seem like a security risk in other contexts (Munroe, 2007), it's probably fine in NetLogo (although there is a `shell` extension . . .). Functionality is limited by the fact that no breeds (NetLogo's version of a class) or global variables can be defined. However, NetLogo has some built-in breeds and variables, and it turns out that it is possible to implement a rudimentary agent-based model entirely through this limited interface between the GUI and the underlying code. The astute reader will immediately have spotted the emergence (ha! agent-based modelling joke) of a possible recursion. If we can embed an agent-based model, can we also embed an entire agent-based modelling tool? Thus, can we put an agent-based modelling tool inside a presentation made with a presentation editor inside a model made with an agent-based modelling tool? I can answer this question with a resounding "Yes, if you squint a bit".

5.2 Proof

I prove the recursive properties of `NetPlop` by embedding a two-dimensional cellular automaton, Conway's Game of Life (Gardner, 1970), in a presentation slide made with `NetPlop` solely through its GUI. The NetLogo code is given below. As the Game of Life is Turing-complete (Rendell, 2002), this means that (in principle) any computational task could be done inside `NetPlop`, including implementing an agent-based modelling tool. I leave this as an exercise to the reader.

```
let gol-patches patch-set patches with [
    pxcor > 100 and pxcor < 200
    and pycor > 50 and pycor < 100
]

ask gol-patches [
    ifelse random-float 100.0 < 35 [set pcolor black]
    [set pcolor white]
]

repeat 100 [
    ask gol-patches [
        set plabel-color [255 0 0 0]
        set plabel count neighbors with [pcolor = black]
    ]
    ask gol-patches [
        ifelse plabel = 3 [set pcolor black]
        [if plabel != 2 [set pcolor white]]
    ]
    wait 0.1
]
```

6 Advantages over Established Presentation Editors

NetLogo and `NetPlop` are free, which may encourage widespread adoption. Also, you can embed agent-based models in your presentation directly, obviating annoying workarounds like writing a Python package to export your NetLogo model runs as `.MP4` files so you can then add them to PowerPoint slides (Steinmann, n.d.).

7 Disadvantages Compared to Established Presentation Editors

Many.

8 Conclusions

I have shown that it is possible to implement a feature-complete presentation editor in NetLogo. This editor has the added feature of being able to embed agent-based models, unlike commercially available presentation editors. As such models can be Turing complete, any given computational task can be completed with them. With any luck, this is a step towards the software Singularity where every program is every other program.

Acknowledgements

Jason R. Wang, Mikhail Sirenko, and Connor McMullen were wise enough to steer clear of this project, but still offered feedback from the peanut gallery. I would also like to thank Igor Nikolic and Martijn Warnier, my professors for Complex Adaptive Systems, for sparking my interest in both agent-based modelling and programming.

Disclaimer

I am not affiliated with, nor is this work endorsed by, the Microsoft Corporation.

Software Availability

Name `NetPlop`

Description `NetPlop` is a presentation editor built entirely in NetLogo. It consists of two NetLogo models, the Editor and the Viewer, used for creating and displaying slide decks. Both models are available through the CoMSES Net Computational Model Library:

`www.comses.net/codebases/?query=netplop`

Developer Patrick Steinmann

Source language NetLogo, NetLogo GUI

Supported systems Anything you can install NetLogo on, but probably not NetLogo Web

License BSD-3

References

CoMSES Net. (n.d.). *Computational Model Library.* Retrieved from `www.comses.net/codebases/` (Accessed 2021-03-26)

Gardner, M. (1970). Mathematical Games. *Scientific American*. Retrieved from `scientificamerican.com/article/mathematical-games-1970-10`

Microsoft Corporation. (n.d.). *PowerPoint.* Retrieved from `www.microsoft.com/en-ww/microsoft-365/powerpoint` (Accessed 2021-03-26)

Munroe, R. (2007). *Exploits of a Mom.* Retrieved from `www.xkcd.com/327/`

Rendell, P. (2002). Turing Universality of the Game of Life. In A. Adamatzky (Ed.), *Collision-Based Computing* (pp. 513–539). London: Springer London. doi: 10.1007/978-1-4471-0129-1_18

Steinmann, P. (n.d.). *aul: A Python package for saving NetLogo runs as media files.* Retrieved from `www.github.com/steipatr/aul`

Wikipedia. (n.d.). *Presentation program.* Retrieved from `www.wikipedia.org/wiki/Presentation_program` (Accessed 2021-03-26)

Wildenhain, T. (2017, April 1st). On the Turing Completeness of MS PowerPoint. In *A Record of the Proceedings of SIGBOVIK 2017* (p. 102-106).

Wilensky, U. (n.d.). *NetLogo.* Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.