AgentBase:
Agent Based Modelling in the Browser

Wybo Wiersma

Oxford Internet Institute
University of Oxford
mail@wybowiersma.net

Abstract. *AgentBase.org* allows for Agent Based Modeling (ABM) directly in the browser. One can edit, save, and share models without installing any software or even reloading the page. Models use the AgentBase Library, and are written in CoffeeScript, which is instantly interpreted as JavaScript. The AgentBase Library provides a rich set of resources for moving and drawing agents, neighbour detection, and other things expected from ABM tool sets. AgentBase is optimized for making illustrative models. It is opinionated software which values simplicity and clean model code over CPU performance.

1 Introduction

AgentBase.org is a minimalist Agent Based Modeling (ABM) platform that allows one to quickly build models that run directly in the browser [1]. It follows NetLogo's Agent oriented Programming model and is entirely implemented in CoffeeScript [2, 3, 4]. Its quick feedback between tinkering and model behaviour makes it surprisingly fun to use. And the ability to share models with nothing more than a hyperlink (no software or browser-plugins to install), across all major browsers (IE, Firefox and Chrome), makes the models very accessible.

After AgentBase.org is described more in depth, the design philosophy behind it will be explained, which will clarify how it relates to other ABM toolsets. This will be followed by a brief description of how to build your first model with AgentBase.org.

2 AgentBase.org and the AgentBase Library

AgentBase.org provides the web-platform, while the AgentBase Library is the toolset for model-building [5]. The toolset is Open Source (GPLv3) and written in CoffeeScript. In layman's terms CoffeeScript is a clearer, less verbose dialect of JavaScript. The AgentBase toolset was derived from AgentScript, a similar Open Source toolset that tries to re-implement NetLogo in JavaScript [6]. The AgentBase Library is more streamlined, and more thoroughly tested than AgentScript, but we will get back to this in the design philosophy section. Besides the CoffeeScript language, that - through JavaScript - allows models to run directly in the browser, HTML canvas is also used to render models in the browser [7]. Models made with the AgentBase library can be exported, and used on any website, not just on AgentBase.org.

The web platform, AgentBase.org, is a site for sharing and editing models. It uses the ACE text editor, which mimics the powerful editors used by programmers, in the browser, and is implemented in JavaScript, so it does not require anything to be installed [8]. The ABM model code held by the editor is instantly (client-side) compiled from CoffeeScript to JavaScript using the *Browser.coffee* library, and then executed, when the user hits run [9]. So changes are immediate. ABM models, and any changes to them are stored on GitHub, the site most commonly used for sharing and managing code repositories [10]. They are stored in so called Gists, which are mini-repositories. Users of AgentBase.org thus need a GitHub account to save models in their own name. No other accounts (not even an AgentBase.org account) are needed. On the server-side, AgentBase.org is implemented in about 150 lines of NodeJS, and it uses the MongoDB database for keeping track of models' Gists repositories [11, 12].

3 Design philosophy

There are two types of ABMs that we define here as Illustrative and Predictive. Illustrative ABM models are models that serve to illustrate a sociological or other process to fellow scholars, or other human audiences. A good example is Schelling's segregation model, which illustrated that even slightly racist preferences can lead to complete segregation on a macro scale, and that thus segregation does not imply high levels of racism, as was previously thought. Yet because it was an illustrative model, and kept simple - it used a chessboard type grid, not real city maps - the model could not be used to predict segregation in actual neighbourhoods [13, 14]. Illustrative models are kept intentionally simple so they can be more effectively communicated. While predictive models approach reality much closer in terms of complexity, making them better at predicting outcomes, but too opaque to illustrate individual processes. The test of an illustrative ABM is not predictive power, but whether it elucidates the modelled process to colleagues; whether they think it make sense [15].

AgentBase is optimized for the quick development of illustrative - not predictive - ABM models. And it is exactly because illustrative models benefit from being easily shareable, that AgentBase was made to run in the browser. Similarly, CoffeeScript, a relatively slow scripting language, was chosen for its readability. The beauty and elegance of a language - so called syntactic sugar - matters a lot for this, as well as for productivity. CoffeeScript is a lot cleaner than JavaScript, because it leaves out semicolons, and superfluous brackets, and makes Object-oriented programming a lot more straightforward. And while it is true that most software engineers are more familiar with JavaScript than with CoffeeScript, illustrative models - and their code - aim to speak to scholars, not to software-engineers, and they will find CoffeeScript much easier to read. Other design-choices in terms of function-naming, etc. that make the model code easier to read, but slightly slower, have been favoured as well. AgentBase values readable, pretty code over CPU performance.

Another choice that sets AgentBase apart, is that unlike Tortoise (and AgentScript), it does not aim to reimplement NetLogo, or parts of it. Though NetLogo is a very rich ABM toolset, many of its functions and some of its design are optimized for the Logo language, and the Desktop environment, and some of it could also be considered bad. Some of its functions are - for example - really badly named for readability, such as 'cp' shortcut for clear patches, and 'fd' for forward. Also, CoffeeScript as a language is quite different from Logo (it is more inspired by the C and Java languages than Lisp), and therefore different ways of organizing functionality make more sense. In addition to furthering the illustrative power of models, readable code that can be changed and shared from a webpage as easily as a Wiki document, also helps lower thresholds to contribution. This would be expected to help more people get involved, similar to how Wikis did this for text-editing. Which could further models' reusability, and increase their academic impact. Finally, AgentBase also sets itself apart because it is well-tested through good coverage by automated tests.

4 Your first model

Models for AgentBase only need to implement two methods: `setup()` and `step()`. The Template Model on agentbase.org essentially consists of the following [16]:

```
class MyModel extends ABM.Model
  setup: ->
    @patches.create()

    for agent in @agents.create 25
      agent.shape = u.shapes.names().sample()

  step: ->
    for agent in @agents
      agent.forward()

new MyModel(div: "model").start()
```

`Setup()` creates the patches and 25 agents, and gives them each a random shape. `Step()` then makes the agents move forward. Visit `AgentBase.org` and tinker with it! (and save it under a different name)

Finally, if you want to embed this model in your own webpage, just download and unzip AgentBase. The model can then be directly embedded in a webpage with:

```
<html>
  <head>
    <script src="agentscript.js"></script>
    <script src="coffee-script.js"></script>
    <script type="text/coffeescript">
    ...your model...
    </script>
  <head>
  <body>
    <div id="model"></div>
  </body>
<html>
```

To conclude, AgentBase is a toolset well-suited for the quick development of models that can easily be shared and extended. Its use of CoffeeScript, makes models more readable, without diverging from browser standards. And though AgentBase limits itself to the niche of illustrative models, this is the niche that many, if not most, academic models fall into.

References

[1] *AgentBase*, 2015. [Online]. Available: `http://agentbase.org/` (visited on 03/29/2015).

[2] S. Tisue and U. Wilensky, "Netlogo: A simple environment for modeling complexity," in *International conference on complex systems*, 2004, pp. 16–21.

[3] *NetLogo*, 2015. [Online]. Available: `https://ccl.northwestern.edu/netlogo/` (visited on 04/25/2015).

[4] *CoffeeScript*, 2015. [Online]. Available: `http://coffeescript.org/` (visited on 04/25/2015).

[5] *AgentBase Library*, 2015. [Online]. Available: `http://lib.agentbase.org/` (visited on 04/25/2015).

[6] *Agentscript*, 2015. [Online]. Available: `http://agentscript.org/` (visited on 03/29/2015).

[7] *HTML Canvas Element*, 2015. [Online]. Available: `http://en.wikipedia.org/wiki/Canvas_element` (visited on 04/25/2015).

[8] *Ace - Code Editor for the Web*, 2015. [Online]. Available: `http://ace.c9.io/#nav=about` (visited on 04/25/2015).

[9] *Browser.coffee*, 2015. [Online]. Available: `http://coffeescript.org/documentation/docs/browser.html` (visited on 04/25/2015).

[10] *GitHub*, 2015. [Online]. Available: `https://github.com/` (visited on 04/25/2015).

[11] *Node.js*, 2015. [Online]. Available: `https://nodejs.org/` (visited on 04/25/2015).

[12] *MongoDB*, 2015. [Online]. Available: `https://www.mongodb.org/` (visited on 04/25/2015).

[13] T. C Schelling, "Dynamic models of segregation," *The journal of mathematical sociology*, vol. 1, pp. 143–186, 1971.

[14] T. C. Schelling, *Micromotives and macrobehavior*. WW Norton & Company, 2006.

[15] J. H. Miller and S. E. Page, *Complex adaptive systems: An introduction to computational models of social life: An introduction to computational models of social life*. Princeton University Press, 2009.

[16] *AgentBase Template Model*, 2015. [Online]. Available: `http://agentbase.org/model.html?9d54597f7aafc995d227` (visited on 04/25/2015).