aima-javascript

interactive visualizations for the book,
Artificial Intelligence: a Modern Approach

introduction

Joel Huang (@joel-huang)
Junior, B.Eng, Computer Science and Design
Singapore University of Technology and Design

landing page	***github***	***linkedin***	***project blog***	***university blog***	***resume***
joel-huang.github.io	@joel-huang	Joel Huang	jankyourflonk	SUTD	pdf

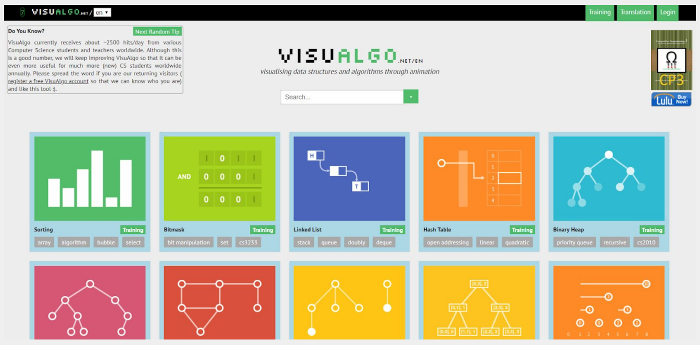
Hi! My name is Joel, and I'm currently a Junior at the Singapore University of Technology and Design. I enjoy working on projects, code daily, and creating things, including graphics, software, sound, games, and electronics. At my University, the art of design thinking is ingrained, sometimes too deeply, in almost every student: it makes for impactful and purposeful products and services we create. I found out about GSoC when Google promoted it at my University, last week (I scrambled to get to work on this!), and aima-javascript stood out as the only project that I *really* want to do this summer: a blend of design, graphics, algorithms and a first step into the world of open-source development. Over the summer, I'll be doing some intern work on weekday afternoons, but I'll be able to spend, on average, 5 hours a day working on aima-javascript. Of course, weekends will be entirely available!

I try my best to vary the type of projects I do, and exercise design *through* these different forms. Having started in 2016, I've done software projects in Java, Python, C#, and Javascript; games in Unity3D and Verilog HDL; personal hardware projects including a overdrive guitar pedal and a mechanical keyboard; and participated in hackathons and game jams (my team is organizing an upcoming hackathon in September too!). A list of my projects is on my project blog, jankyourflonk. Some of them might also be on GitHub at @joel-huang. In all my projects I try to design for user experience, for art as well as function, and ultimately with purpose and with substance. I took a look at the recommended visualization literature (@redblobgames), and I can see the thought process that goes into designing the different games. I fully agree that visualizing algorithms should *be* a game, ultimately giving context and life to an algorithm that is just a bunch of lines and instructions.

proposal

design philosophy | *intuition through interaction*
Visualizations need to add value to the learning process. They should not replace the printed source material, but supplement the material to build intuition through additional sensory input(s) (sight, sometimes touch and maybe even sound). The learner, by observing and playing with the visualization, should receive feedback, often subconsciously: (i) how does this visual model *react* to what I change? (ii) what *makes* it react in this particular way?

strategy & communication | *agile*
strategy | Agile development! Sticking with the quality over quantity mantra, target 10 implemented algorithm visualizations over the entire summer, starting with simple, familiar algorithms and cracking the more complex ones (complexity here means a more thoughtful design process is needed) later. The expected results should be re-evaluated before every phase of mentor evaluation (total of 3 times). Having access to hardcopy editions (I have a copy with me now!) through my school library will be particularly convenient as I always appreciate a print copy of materials I work with, especially literature or research materials. The specific parts of the book I'm most interested in implementing are Part III: Knowledge Reasoning and Planning, and Part V: Learning.
communication | I plan to send a gist/review every week, with additional communication as necessary. I believe communication is the best way to build a great product in open source, so I'll be getting in touch with the mentor often. I am comfortable with working with git, and on github issues, google docs, jupyter notebook, telegram, slack, discord or any platform that best serves the project.



Do You Know?
VisuAlgo currently receives about ~2500 hits/day from various Computer Science students and teachers worldwide. Although this is a good number, we will keep improving VisuAlgo so that it can be even more useful for much more (new) CS students worldwide annually. Please spread the word if you are our returning visitors (register a free VisuAlgo account so that we can know who you are) and like this tool :).

proposal | *landing page and navigation revamp*

While aima-javascript needs concrete implementations for visualization, and I would love to work on those, I would also like to propose a concurrent re-structuring of the site, or at least the way the navigation system is implemented. I found the current site (using bootstrap) a little time-consuming to maneuver and it is not easy to retrieve a specific visualization - a pity since the visualizations are really great. A dashboard landing page with an immediate search might make for less hassle in the learning experience by abstracting the directory structure away: Visualgo.net is one such inspiration.

proposal | *part III: knowledge reasoning and planning*
Looking through all the chapters briefly, designing logical visualizations are difficult, i.e. the most amount of planning and testing is needed, because teaching logic visually often requires positing a scenario, an environment, and different agents. I would think the designer needs to tread the thin line between abstraction and concrete examples, and be open to using interaction as a teaching tool. In the Internet Shopping Example (Section 12.6 of Knowledge Representation), the reader is taken on a long walk through the process of string parsing, link following, and specification comparison. While we may not be able to train our own NN just to demonstrate the following situation, we can mock the outputs and treat each program like a black box; each section can be separated into it's own bite-sized visualization to emphasize the *function* of each layer, that is, what is the *intelligent agent* trying to handle the knowledge *encoded* in the input it receives? A drag-and-drop input:receiver visualization might work here, as I can try many varying, possibly random, inputs and see how the receiver responds at that layer. **Through doing this a couple of times, I build my intuition** of what the receiver does to the input by looking at an output. It makes us sound like the machines, which is an idea I find pretty interesting :)

proposal | *part v: learning*
In general the visualizations for decision trees, regression, SVMs and neural networks will be straightforward as there are many documented and already visualized methods to draw inspiration from. However, I would like to focus on researching how to best explain backpropagation and activation functions as they are commonly misunderstood, or not understood at all, by beginners. I attended a few early sessions of AISaturdays, hosted by nurture.ai, a Singaporean startup focused on building AI communites, and there was a particular method of visualizing backpropagation that I remember very clearly. As a beginner, going through the entire feed-forward and back-propagation cycle once was immensely helpful in understanding the flow of logic. Else, I would have initially disregarded it as a pile of black-box linear algebra that didn't make sense. I would expect this section to be best implemeted through traditional graph structures (NNs) and scatter plots (regression/SVM), which are good exercises in d3.js but not extremely complex.

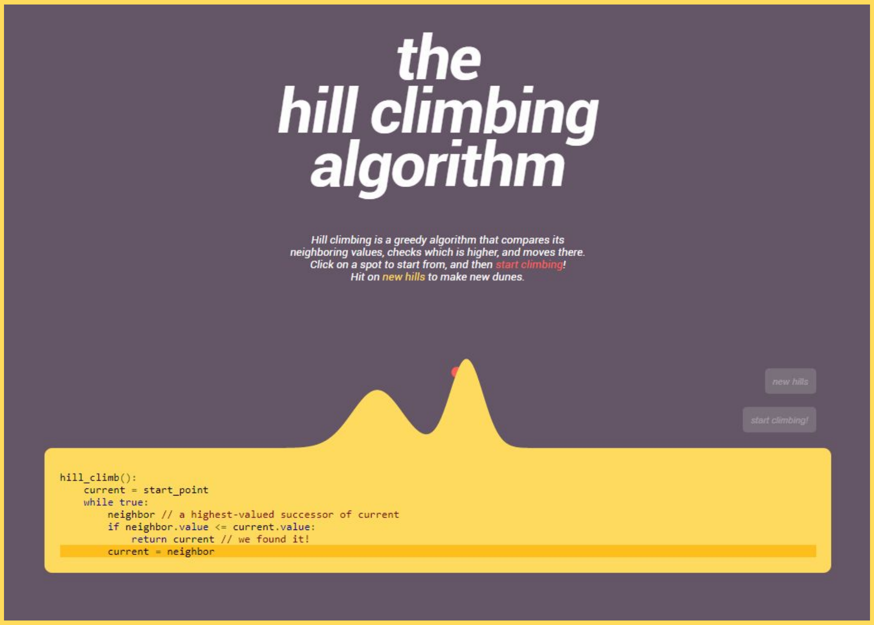
workload
Over the summer I'll be able to commit about 30+ hours per week, with more details in planned timeline below. I'm a meticulous worker and somewhat of a perfectionist so 30 hours might be a lower bound.

timeline

0h/day	1-2h/day	3-4h/day	5-6h/day	6+h/day
⬜	🟩	🟨	🟧	🟥

Date Starting	GSoC Timeline	My aima-javascript timeline	Work intensity
< March 28th	Write Proposal	Prototype and think about how the three months of code can best contribute, and sustain aima-javascript even post-GSoC	🟩
March 28th	Applications close	Get familiar with AIMA, work with with **d3.js** on **one major ongoing project** and pick up **one additional library**, possibly chart.js or plotly.js	🟨
April 23rd	Students announced	University: Finals week	⬜
April 27th		Re-organize site **if possible**: landing page should showcase the visual nature of the project (est. 25-30 hours) 8. First Order Logic (30 hours) ELSE 8. First Order Logic (30 hours) 9. Inference in First Order Logic (30 hours)	🟧
June 15th	Checkoff 1	9. Inference in First Order Logic (30 hours) 10. Classical Planning (30 hours) 11. Planning and Acting in the Real World (30 hours) 12. Knowledge Representation (35 hours)	🟥
July 13th	Checkoff 2	18. Supervised Learning (35 hours) 18. Models (35 hours) 20. Probabilistic Models (30 hours) 21. Reinforcement Learning (30 hours)	🟥
August 5th		Buffer time for bug fixing, code cleaning and testing.	🟨
August 14th	Coding ends		⬜

design document



Hill climbing is a greedy algorithm that compares its neighboring values, checks which is higher, and moves there. Click on a spot to start from, and then start climbing! Hit on new hills to make new dunes.

```
hill_climb():
    current = start_point
    while true:
        neighbor // a highest-valued successor of current
        if neighbor.value <= current.value:
            return current // we found it!
        current = neighbor
```

***prototype** | the hill climbing algorithm*
https://joel-huang.github.io/aima-visualization/hill_climbing.html

The hill climbing algorithm is fairly simple - where you are, check left, check right, see which neighbor is higher, move there, and repeat till the neighbor is greater than or equal to your height.

Since it is a simple algorithm, I juxtaposed the pseudocode against the visualization. The user can see exactly which instructions are being executed as the program runs, and how this *translates* to climbing the hill in the visualization.

For reinforcement, the user can generate new hills, as well as click anywhere on the hills to change the starting point. This functionality encourages users to run the simulation more than once, which is great for building intuition.

design process
saturday, 24 march 2018
So, Google just dropped by my university to introduce GSOC with six days left (now three) till the proposal deadline. I haven't any chance to connect early or send drafts, so walkthrough of my design process and a prototype might be the best way to show my work. I want to create something entirely from scratch, without using any templates.

- Objective: Design a visualization method that conveys AND cements intuition of the algorithm, while being pretty
- Constraints: 2 days work, substance over quantity
- Testing: Input testing, feedback from users (beginner, intermediate, advanced)
- **Strategy:** Choice of algorithm - Hill Climbing (straightforward, familiar, well-documented), Choice of framework - d3.js, js
- **Ideation:** Explaining hill climbing to someone is straightforward, so the visualization needs to add even more value, else there would be little point. We can show the process of decision making at every step through comparing the executed steps in a pseudocode block next to the visualization

11:30pm - created empty repository aima-visualization, git init
12:12am - sketched out layout – visualization side by side with pseduocode

sunday, 25 march 2018
3:09pm - In Hill Climbing it's important to juxtapose local and global maxima, for if a wrong start point is picked, Hill Climbing can miss out on global maxima which are actually nearby.
What is a simple curve to demonstrate?
6:00pm - Completed base HTML/CSS, designed to be visually loud, centered, focused
8:45pm - Prototyped algorithm in javascript, using two summed gaussian distributions of 2000 individual random data points
11:39pm - Picking up new d3.js-fu as I go along is fun and frustrating, completed d3.js transitions for hills and pseudocode highlights

monday, 26 march 2018
01:37am - Changed the color palette to a less saturated one as it was too distracting
03:45am - Having some problems with mapping screen space to data points, refactored some code and wrote out data-to-coordinate functions to clear things up
05:15am - Final touches done, moderate input testing, edge cases handled with disabling of buttons on simulation

post-prototype review

- Yes, this is how I wanted it to look! Imagining myself in the shoes of a high school/first-year/beginner CS student, I can understand this algorithm with the little explanation it provides. The pseudocode is a big part of this, as I'm able to see the statements triggering and the conditional not triggering at every step.
- I often build to exactly how I envision the final product, and it always ends up taking a very long time. Next time, work can be sped up if I re-evaluate the prototype, especially visually, at set checkpoints in the workflow.
- The workflow could be improved slightly as I spent a long time debugging, but was necessary to de-rust my d3.js skills!
- The most problematic part of this was the mapping between SVG coordinate space and data points. Learning point: use x.invert() and y.invert() to return range inverse.