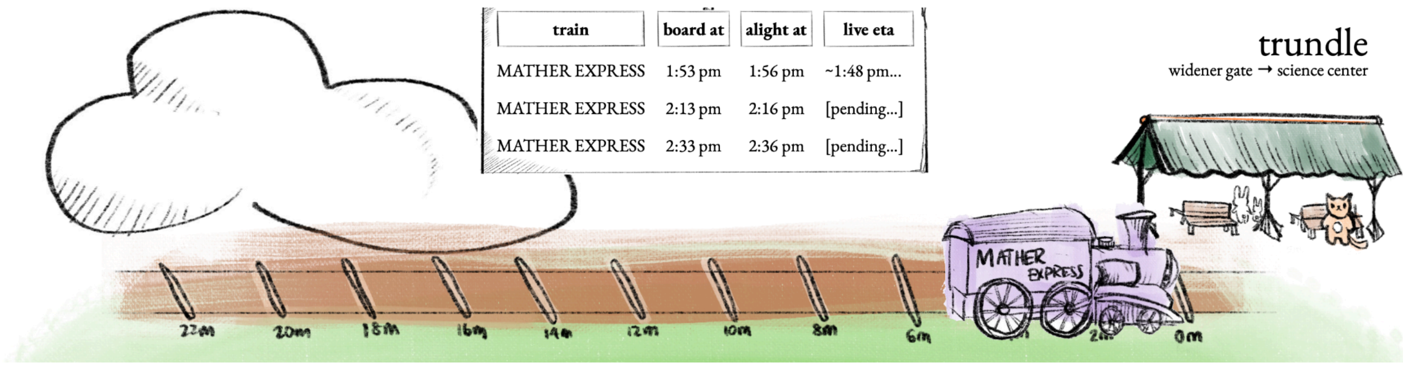
Trundle: the Locomotive-Based Harvard-Shuttling App with Live Data and Happy Animals

Kyra Mo and Patrick Thornton



train	board at	alight at	live eta
MATHER EXPRESS	1:53 pm	1:56 pm	~1:48 pm...
MATHER EXPRESS	2:13 pm	2:16 pm	[pending...]
MATHER EXPRESS	2:33 pm	2:36 pm	[pending...]

Trundle and You

Are you familiar with the Harvard shuttle layout, and just want something to tell you which shuttle to board and when? Do you find the kiosk at the SEC's eastern entrance to be discomfiting and distressing? Would you appreciate a certain degree of cuteness and charm hardwired into your nonetheless quite functional traveling app?

Well, thanks for filling out our questionnaire. While you're here, let us introduce you to **Trundle**. It's an interface for the Harvard shuttle system that leverages abstraction and lovely hand-drawn art to allow you immediate access to the information you need most: what's coming, how soon, how sure we are about that.

Let Me See

Source code: here! Demonstration video, which leisurely tours through some crack-of-dawn Trundle usage: here!

Conceptual Underpinnings

An application of this scope is necessarily going to involve a whole slew of concepts working in tandem. We're pressed for space here, though, so we'll focus on one that we believe is at least very near to the core of Trundle: the *suggestion* concept.

Its **purpose** is to allow users to quickly grasp 1) what their best options are for getting from a start stop to a destination stop and 2) information about those options, such that one can make a reasonable decision between them in a hurry.

We'll get to the technical implementation details of this concept in a moment, but more immediately pertinent to Trundle are the ways in which we elected to abstract and aestheticize certain aspects of this concept in the interface to imbue the correct amount of uncertainty in the user.

For one (and this is a big one), we represent each *suggestion* not only as a row in the suggestions table at the top of the page with quantitative information, but also as a water-colored **train** that trundles along its tracks. The tracks act as an aestheticized timeline; the transverse bars[1] demarcate two-minute intervals, and the train station at the right (complete with bunnies and a cat) represents arrival. While there may be more than three suggestions for a given pair of start and destination stops, we show at most three trains to provide an immediate and tangible visual intuition for the user. This way, they'll know at a glance their options and how much urgency is warranted.

Shuttles are notoriously difficult to predict; we afford the user an understanding of the **uncertainty** inherent in the shuttle system in a number of ways. First, we provide live ETA data at the rightmost column of the suggestions table, which pulls info from the tripUpdates.json route provided from passioGo. (On the linguistic layer, a simple tilde affords some preliminary uncertainty, and the animated ellipses keeps users aware of the live nature of the data.) Second, the trains provide a sneakier means of conveying uncertainty; namely, their width over the timeline implies a range estimate, rather than a point estimate, for arrival time.[2] In other words, each train occupies about four minutes of space on the timeline, suggesting an equivalent amount of wiggle

[1] These are properly called *railroad ties* or *crossties* in American English or *sleepers* in British English.
[2] This is similar to the 'When-ish is my Bus' box plot option, except Trundle doesn't purport to be a study on bus-boarding accuracy; we cater to a wider audience, unfamiliar with statistics.

room in our ETAs. Finally, the aesthetics of Trundle (i.e. the cutesiness) implant a certain degree of uncertainty within the expectations of the user at a subliminal level. Shuttles are maddening and mercurial by nature; an app interested in conveying the maximum amount of truth about them ought to be candid about what it cannot guarantee in its very first impression.

Just as much as what we chose to do, Trundle is defined by what it purposefully omits: we omit live position data for shuttles, for instance, and we use a hand-penciled map only for stop selection and hide it afterward. These omissions are a result of our choosing a particular **audience** for this application: namely, Trundle can act as a perfect replacement for the kiosks installed at certain locations in Harvard campus, or as an app for a user already familiar with the shuttle layout. In addition, we avoid a lot of the When-ish-is-My-Bus-style heavy-duty statistical displays, which we feel aren't conducive to rapid understanding; we've simplified to an uncertain point estimate in the table and a corresponding range estimate through the body of the train. We feel these simpler, stylistic methods provide a more user-friendly and intuitive answer to the question posed by 'When-ish is my Bus': representing uncertainty in shuttle ETA.

Nuts 'n' Bolts

Trundle is implemented in **Svelte**, the web development framework. Compared to our previous project, the Output Project comprises a new degree of complexity; thankfully, we found Svelte to provide a commensurate degree of sophistication that allowed it to meet the task.

The interrelation of **components** became a particular focus during this project; the trains, table, and map all reside in different components. To share values between them, we used Svelte's concept of *binding* when appropriate, and found an especial place for Svelte's *stores*, which any component at any level of depth can read from or write to. Our application subdivides into two pages; the *map page* and the *trains page*. Nonetheless, Trundle is a single-page application; the contents are simply redrawn to execute the page change, which Svelte makes easy with a simple $map boolean store and conditional rendering via {#if}. Flask does not support this workflow easily; it encourages rerouting.

In addition, Svelte's robust **reactivity** was as helpful as ever; updating both the table and the position of the train in light of live-updates was a breeze.

Trundlers Love Trundle

Playtesting proved to be a wonderfully edifying experience. Certain blind spots in our design became immediately apparent and have been repaired for the version of the application you see now: for instance, the '2m', '4m' markings along the tracks were meant to convey the abstraction from physical distance to temporal distance, as 'm' stood for minutes: but most thought it meant meters, which torpedoed the whole thing. A simple 'minutes' at the right end of the tracks sufficed to patch the issue up.

Other comments plucked at random from playtesting day: many users found it inconvenient that the only way to restart Trundle from the map page was with a refresh, so a quick and seamless 'back' button was added at the upper-left corner: the columns used to read 'board at' and 'arrive at', where 'arrive' was perceived to be ambiguous (arrive at the start stop or the destination stop?), so it's been changed to 'alight at', to mirror the physical imperative of 'board at'; certain suggestions for correspondingly cute audio, which we hope to add in a future update; and Trevor was furious the cat was never actually picked up by the uppermost locomotive, so we've fixed that by making the cat an **amateur trainspotter** by adding **binoculars**.

The Minds Behind It

Trundle is the creation of Kyra Mo and Patrick Thornton, who literally made the whole thing in a string of multiplayer Zed coding sessions (Zed is this nifty new code editor that has VSCode-Live-Share-esque functionality built in); how's that for collaboration?

Kyra thought of the approach, implemented the map functionality and the reactive train movement, and drew all of the art by hand; Patrick reckoned with the static and live data, clarified the *suggestion* concept, and implemented the suggestion functionality and the stop positions on the map.