

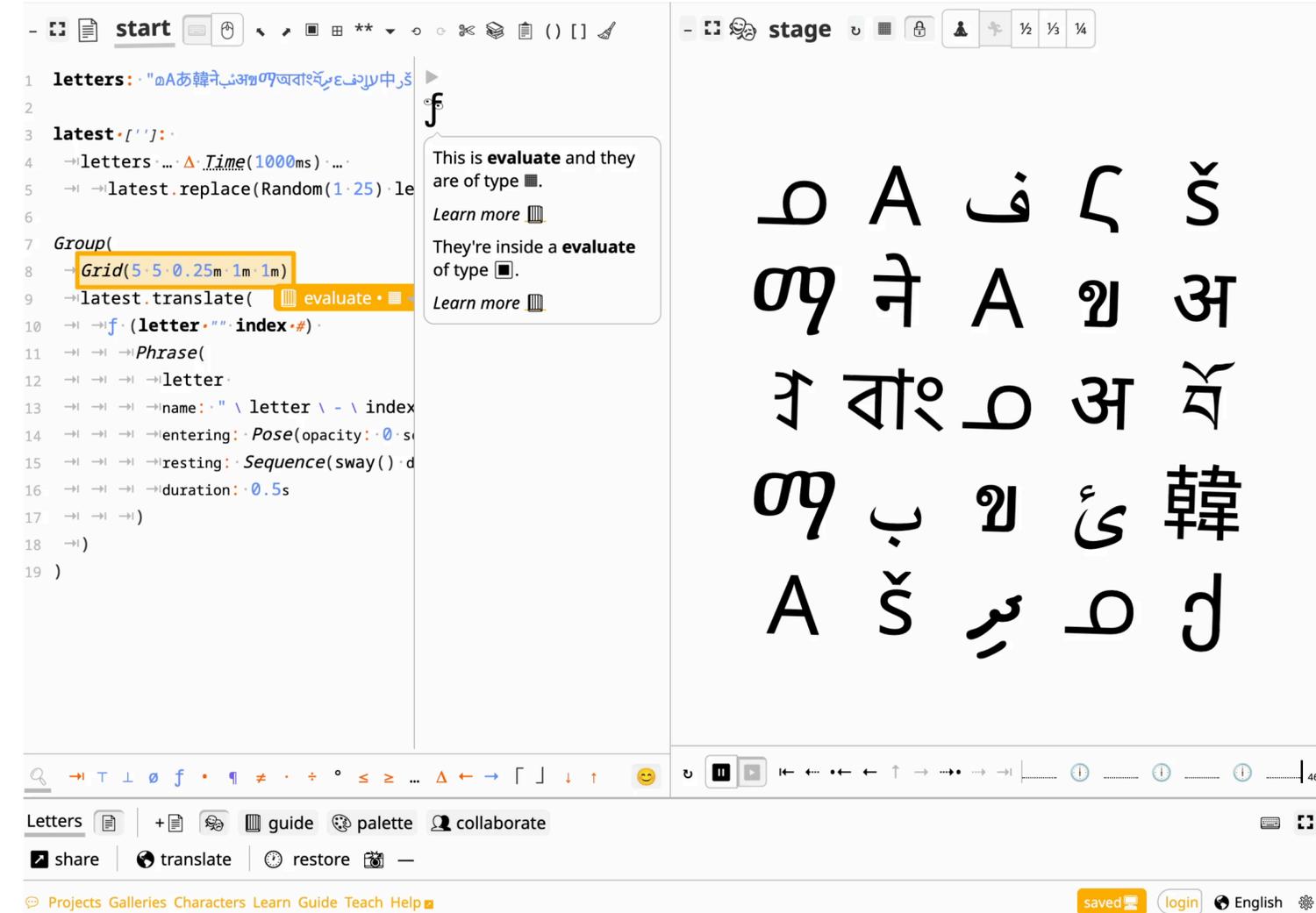
Wordplay: Accessible, Multilingual, Interactive Typography

Amy J. Ko, PhD, University of Washington

Carlos Aldana Lira, Middle Tennessee State University

Isabel Amaya, University of Washington

and featuring... **Adrienne Gifford**, middle school CS teacher extraordinaire!



Programming languages (PL) are a foundational way people interact with computers. But learning them is hard...

They often aren't designed for **learning**.

They are hard to teach, because programming is a **complex skill**.

They are often used to teach CS in ways that **exclude** youth, marginalizing them by their identities.

Two forms of PL marginalization are particularly egregious...

Language code コード شفرة código ኮድ 代码 kodo

PL are often **English- and Western-centric**, requiring English learning before CS learning and ignoring the fluidly multilingual, multicultural world.

Ability      

PL are often **inaccessible**, with syntax, tools, and docs that are not screen-readable, ignore neurodiversity, or require fine-motor movements, like pointing.

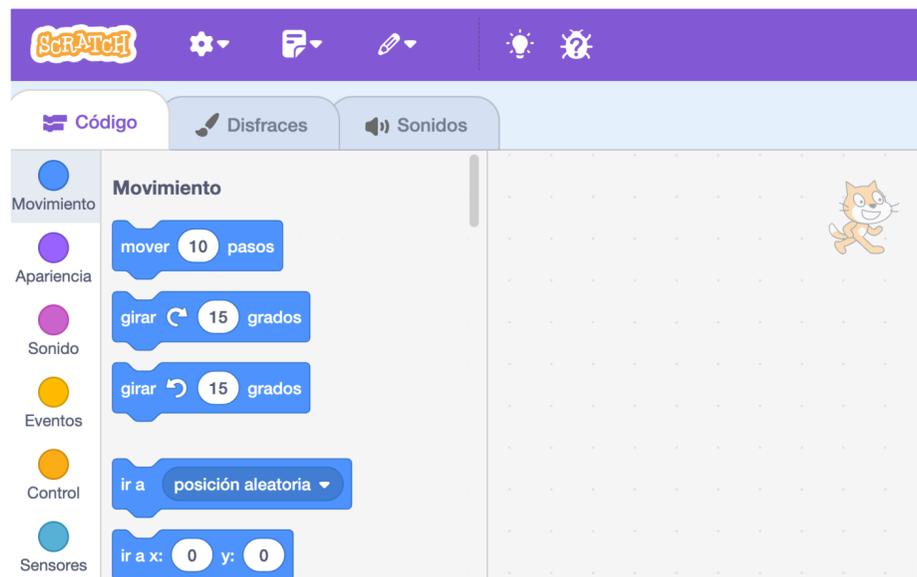
Prior work has made some progress on these two inequities, but in isolation.

Language

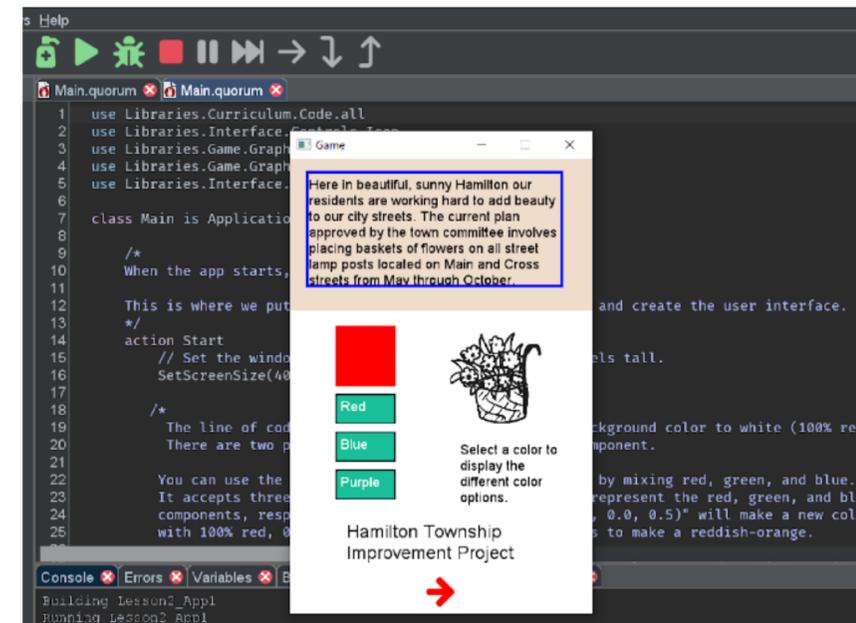
e.g., *Scratch* can be localized to non-English languages, but is inaccessible.

Ability

Quorum is designed to be highly screen readable, but is English only.



Can't use a mouse?
Can't use *Scratch*.



Can't read English?
Can't use *Quorum*.

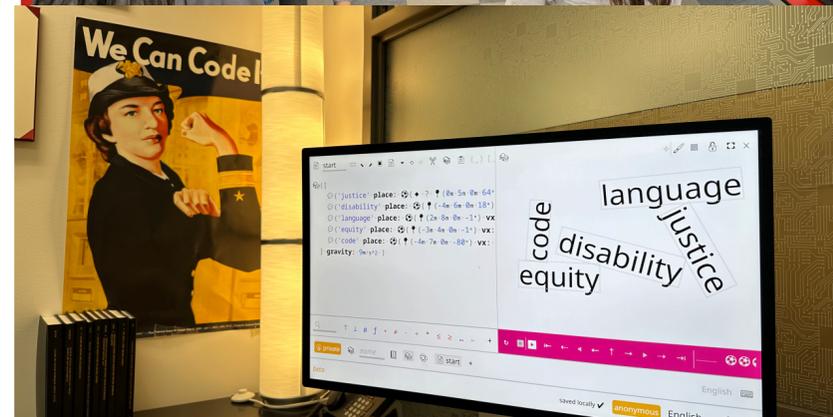
Can educational PL be designed to be both **multilingual** and **accessible**?

What **design, engineering, and pedagogical** challenges arise in designing at this intersection?

e.g., is it just a matter of WCAG compliance and monolingual localization, or are there deeper tensions between the two?

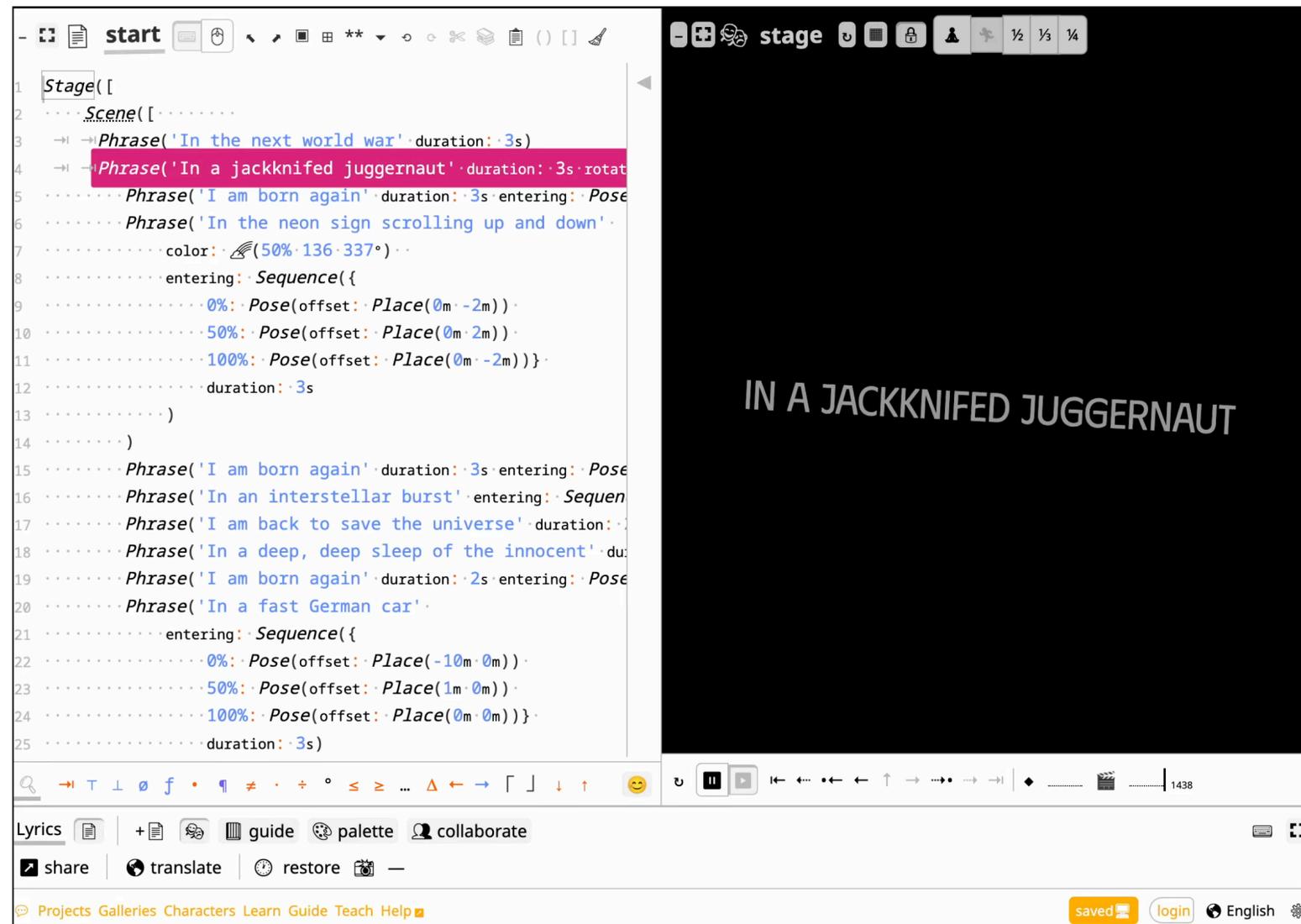
Our approach

- 30 months of community-engaged PL design and development:
 - Partnering with **teachers** to understand the challenges of teaching multilingual youth and youth with disabilities.
 - Engaging **multilingual youth** and **youth with disabilities** to understand tensions with existing multilingual and accessible platforms.
 - Consulting with other **educational PL designers** on how they managed language and accessibility.
- To evaluate our progress and surface new challenges, we convened a summer **focus group** of youth and teachers.



Wordplay!

try it at wordplay.dev



The screenshot displays the Wordplay programming environment. On the left is a code editor with a dark theme, showing a scene definition for a stage. The code includes several phrases with associated durations and entering animations. The phrase "In a jackknifed juggernaut" is highlighted in pink. On the right is a stage view with a black background and the text "IN A JACKKNIFED JUGGERNAUT" displayed in a light gray, sans-serif font. The interface includes a top toolbar with various icons, a bottom toolbar with search and editing tools, and a footer with navigation links and a language selector set to English.

An educational programming language for creating interactive, multilingual, accessible typography.

Let's briefly consider **5 major design** choices, and then hear what youth and teachers think about them.

1. Output is typographic

- Most educational PL use **graphics**, which are inaccessible without image descriptions, and image descriptions are usually monolingual.
- Wordplay, in contrast, uses **text**, which can be web accessible and engage culture and identity.
- Programs can generate multilingual text output, building localization into the PL itself.
- Text is also very low overhead, minimizing internet bandwidth requirements.

Click for the orange shirt (named Waldough) and click on him to win!
Score: 0

4 ADJECTIVES PLEASE!
(SEPARATED BY SPACES)

YOU AWAKEN IN A DARK, COLD ROOM THAT SMELLS OF MOLD.
1. ROLL OVER
2. STAND UP

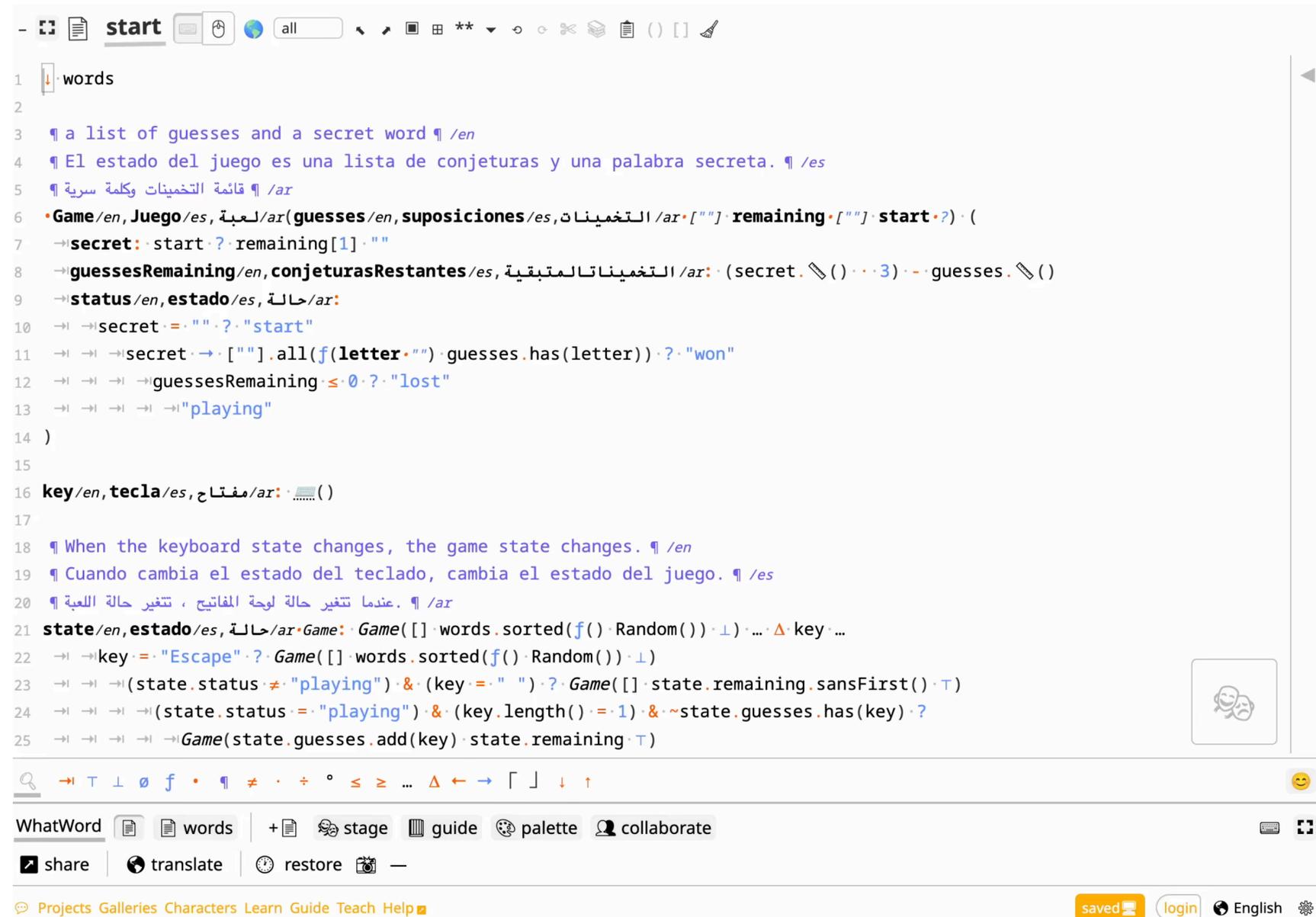
WhatWord

press space to begin

Projects Galleries Characters Learn Guide Teach Help saved login English

2. Code is symbolic and multilingual

- Using **keywords** in a PL embeds a particular language in all programs.
- Wordplay only uses punctuation and multilingual text, allowing youth to embed multiple translations of **text**, **documentation**, and **identifiers** in code.
- The editor can screen read all punctuation for accessibility.
- The editor can translate code into other languages, and render code in one or more languages for multilingual reading.



```
1 words
2
3 ¶ a list of guesses and a secret word ¶ /en
4 ¶ El estado del juego es una lista de conjeturas y una palabra secreta. ¶ /es
5 ¶ قائمة التخمينات وكلمة سرية ¶ /ar
6 •Game/en, Juego/es, لعبة/ar(guesses/en, suposiciones/es, التخمينات/ar·[""]·remaining·[""]·start·?)·(
7 →secret·start·?·remaining[1]·""
8 →guessesRemaining/en, conjeturasRestantes/es, التخميناتالمتبقية/ar::(secret·()·3)·guesses·()
9 →status/en, estado/es, حالة/ar:
10 →secret·="·"·?·"start"
11 →secret·→[""]·all(f(letter·"")·guesses.has(letter))·?·"won"
12 →guessesRemaining ≤0·?·"lost"
13 →"playing"
14 )
15
16 key/en, tecla/es, مفتاح/ar:·()
17
18 ¶ When the keyboard state changes, the game state changes. ¶ /en
19 ¶ Cuando cambia el estado del teclado, cambia el estado del juego. ¶ /es
20 ¶ عندما تتغير حالة لوحة المفاتيح ، تتغير حالة اللعبة ¶ /ar
21 state/en, estado/es, حالة/ar·Game:·Game([ ]·words.sorted(f()·Random())·1)·...·Δ key·...
22 →key·="Escape"·?·Game([ ]·words.sorted(f()·Random())·1)
23 →(state.status·≠·"playing")·&·(key·=" ")·?·Game([ ]·state.remaining.sansFirst()·T)
24 →(state.status·="playing")·&·(key.length()·=1)·&·~state.guesses.has(key)·?
25 →Game(state.guesses.add(key)·state.remaining·T)
```

WhatWord | words | + | stage | guide | palette | collaborate

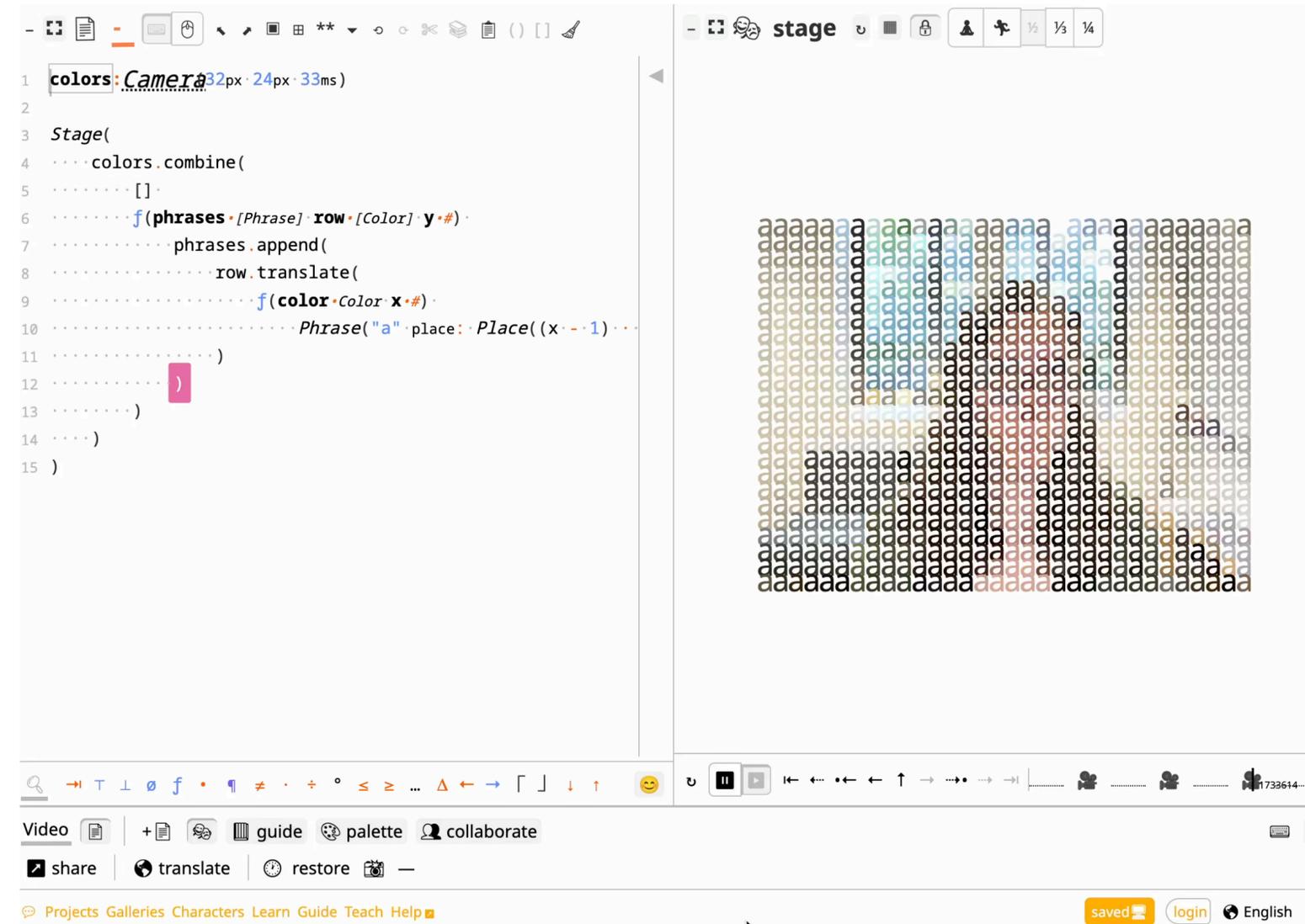
share | translate | restore |

Projects Galleries Characters Learn Guide Teach Help

saved | login | English

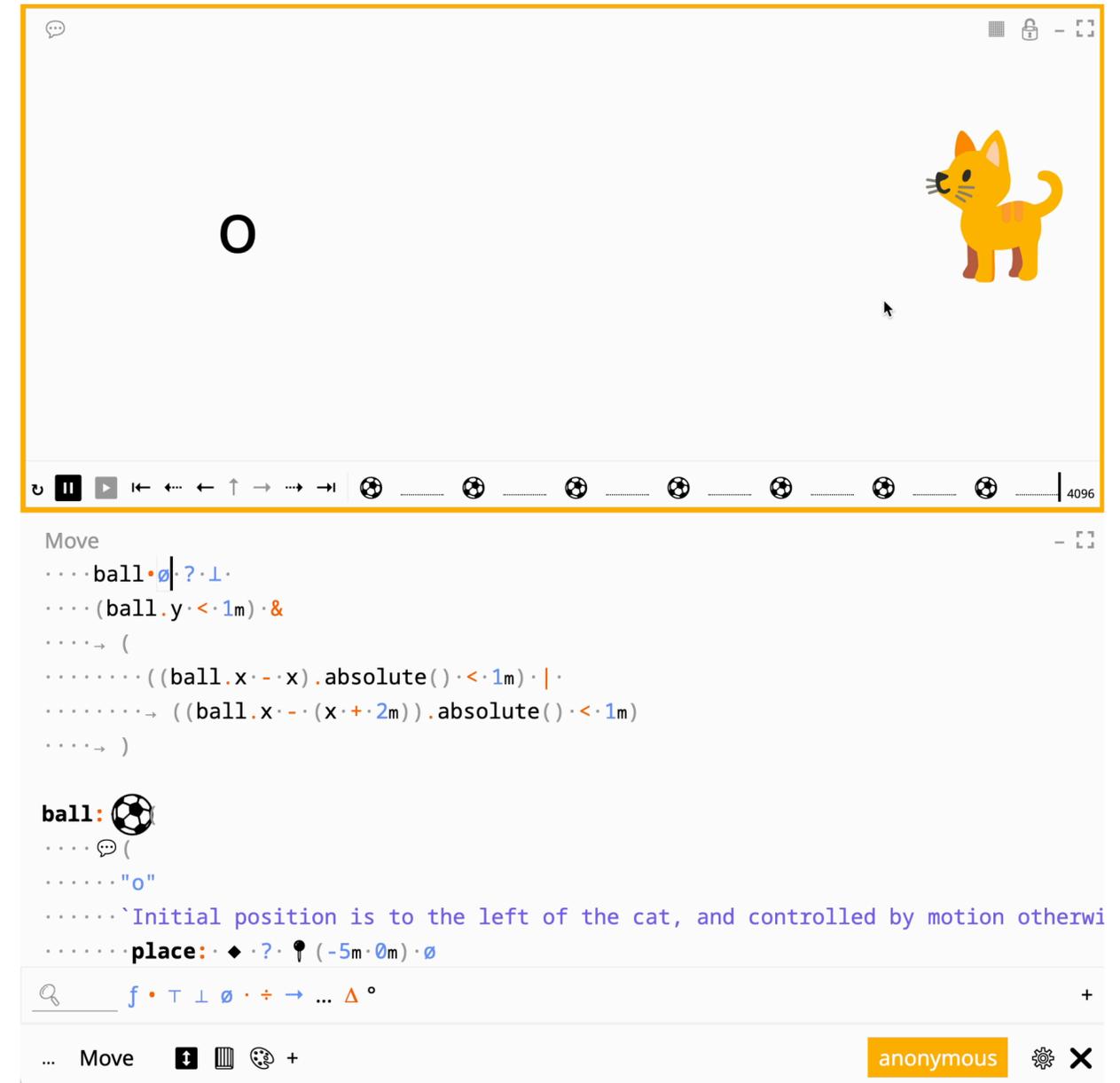
3. Code is functional and reactive

- Imperative code is verbose and does not explicitly describe its purpose, potentially creating friction in screen reading.
- Wordplay is **functional** and **reactive**, so its code is declarative, shorter, and achieves rich interactivity with minimal code, indirectly benefitting accessibility and language inclusion by reducing syntactic complexity.
- This example translates a stream of pixels from a camera into a typographic grid of colored letter a's, with just a few function calls.



4. Accessible, multilingual time travel

- Debugging tools are often monolingual and inaccessible.
- Wordplay allows for **instant, random access, multilingual, WCAG-compliant** time travel through program evaluation history.
- This is possible only because of its purely functional nature: it's only necessary to record a history of inputs, and the rest can be recomputed.



5. Governance is student and teacher led

- The project is open source, free, and facilitated by researchers.
- Teachers and youth, however, set design priorities, and curate design feedback.
- An undergraduate community of open source contributors help realize their feedback in a community of peer learning and teaching.



Evaluation

- These and other design choices are not necessarily good design choices.
- To evaluate them, Carlos and Isabel, undergraduate community members convened a **focus group** of 2 teachers, 3 middle-school students, and 1 undergraduate to examine the platform's design tradeoffs.
- We held **5 sessions**, asking the group to examine Wordplay relative to other educational PLs, using the design principles that emerged from our design process.

What are we doing today?

1. Review guidelines for these sessions
2. Talk about what *Accessible* means
3. Discuss how educational programming languages can be *Accessible*
4. Evaluate Wordplay by the *Obtainable* requirement
5. Reflect on our work

7 design principles from the community

We structured the sessions around 7 principles that had emerged from our 30 months of design work with teachers and students.

The community believed educational PL should:

- **Accessible** — be usable by all students with whatever abilities they have.
- **Liberatory** — be about the “what” and “why” of computation, not just “how”.
- **Transparent** — offer debuggers that fosters self-efficacy for how programs work.
- **Cultural** — enable teaching that is responsive to students’ identity.
- **Obtainable** — be free and compatible with the devices and internet they have.
- **Democratic** — be led by students and teachers to shape the platform’s design.
- **Enduring** — be supported long term to enable long-term investments in teaching.

Focus group critiques

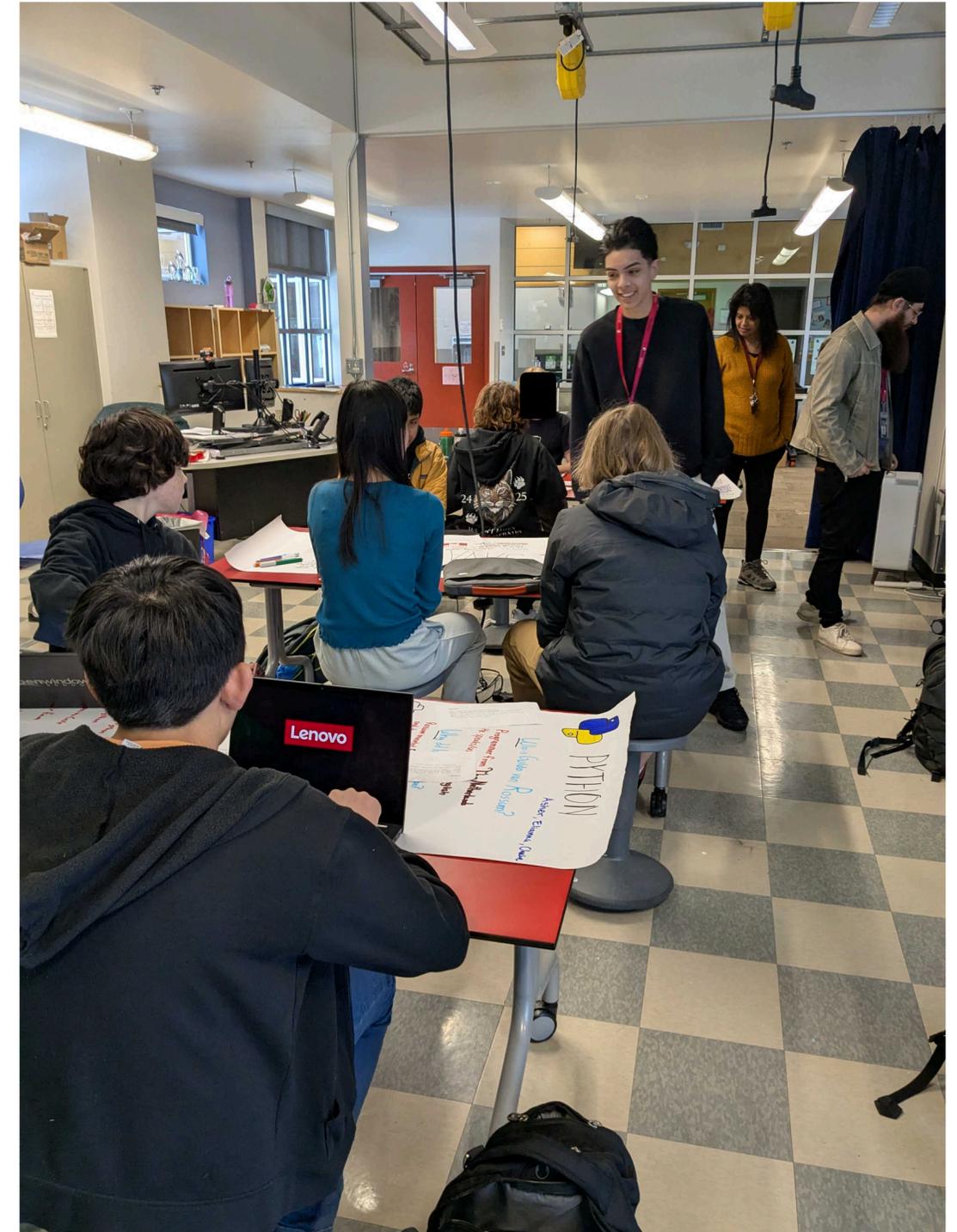
✓ The group affirmed the goals of being web-based, screen-readable, usable offline, broadly multilingual, but also noted several gaps in Wordplay's multilingual accessibility.

✗ It's only partially usable without internet access	✗ Youth are unsure about the benefits of functional code
✗ Smartphone layout and speed is inadequate	✗ It needs to support multilingual collaboration
✗ It needs more customizability for access needs	✗ It should frame youth as shaping future of computing
✗ It needs more diverse youth governance	✗ Not using keywords may disadvantage everyone
✗ Youth can't use GitHub due to privacy policy	✗ There should be multiple ways to learn the PL

Tensions between accessibility and language

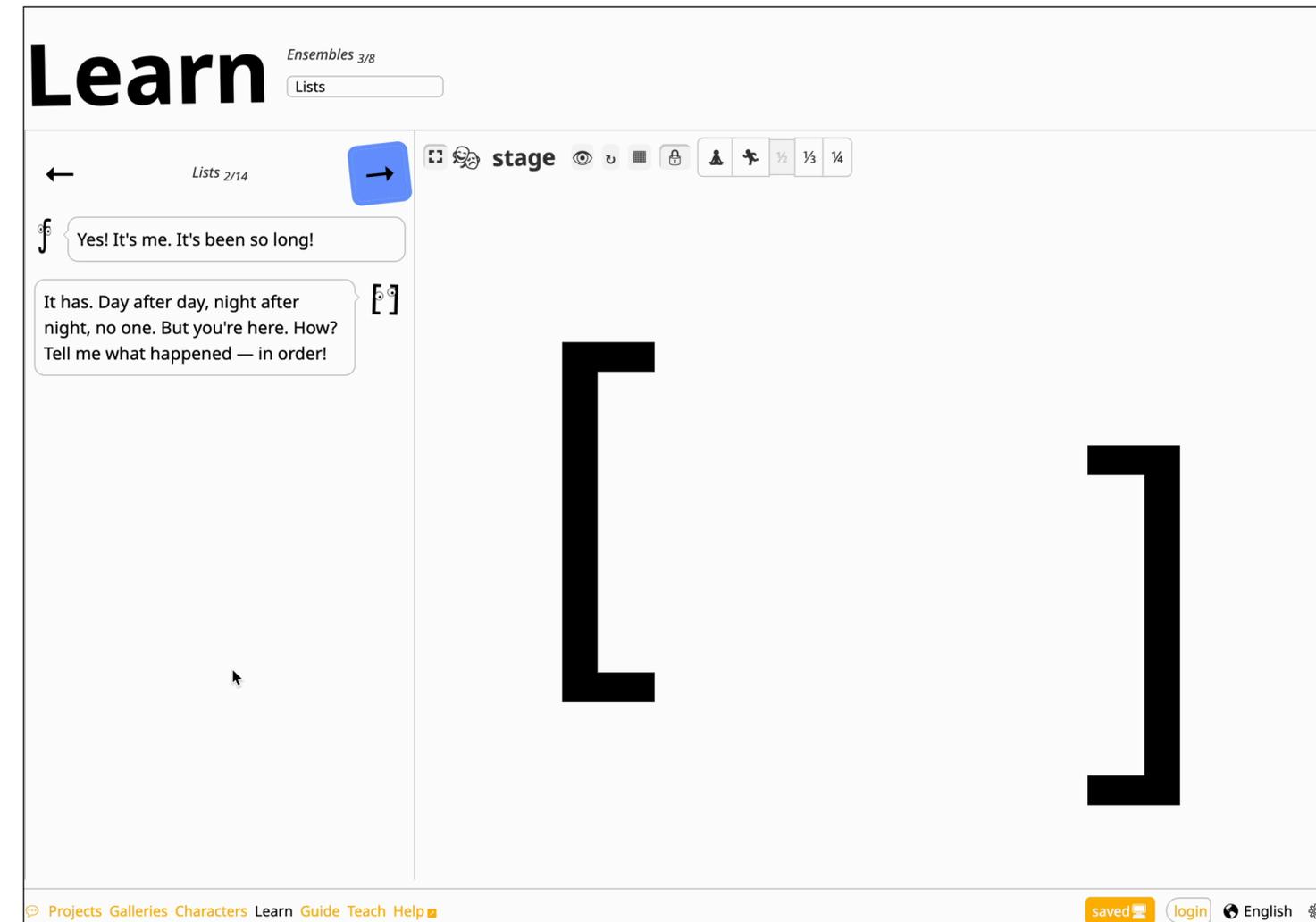
Aspiring for both accessibility and language inclusion may trade off against simplicity, make PL **more complex** to build, learn, configure, use and teach:

- If everything has a description for access, everything must have **thousands of descriptions** for every language and reading level.
- Focusing on text may increase accessibility and language inclusivity within the domain of interactive type, but at the expense of **other kinds of culture and identity** (e.g., music, imagery).
- **Balancing power** through youth governance of an open source project raises numerous tensions between self-efficacy and expertise.



Key insights

- It is possible to build an accessible, multilingual educational PL.
- Doing that universally surfaces real tensions between youth, teachers, and their language and access needs. Everything required to overcome those tensions — diverse, sustainable governance, engineering agility — require resources that EPLs often do not have.



Amy J. Ko, PhD, University of Washington

Carlos Aldana Lira, Middle Tennessee State University

Isabel Amaya, University of Washington

A big thanks to our teacher partner **Adrienne Gifford**, and to the National Science Foundation! This work was supported by grants 2318257, 2137312, 2122950, and 2031265.