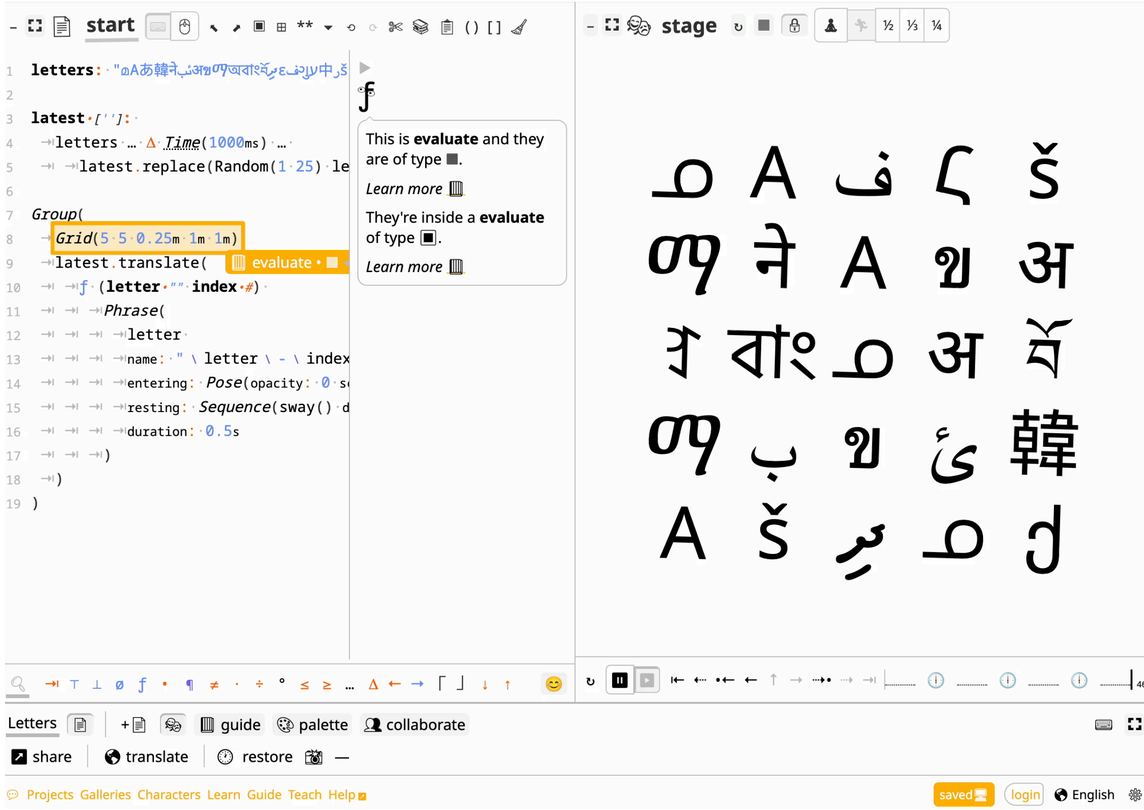
Wordplay: Accessible, Multilingual, Interactive Typography



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!

Amy

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.

Carlos

Two forms of PL marginalization are particularly egregious…

Language code コード شفرة código ክፍ 代码 kodo

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

Ability ⌨️ 🖱️ 🎤 🗣️ 👆 🦽

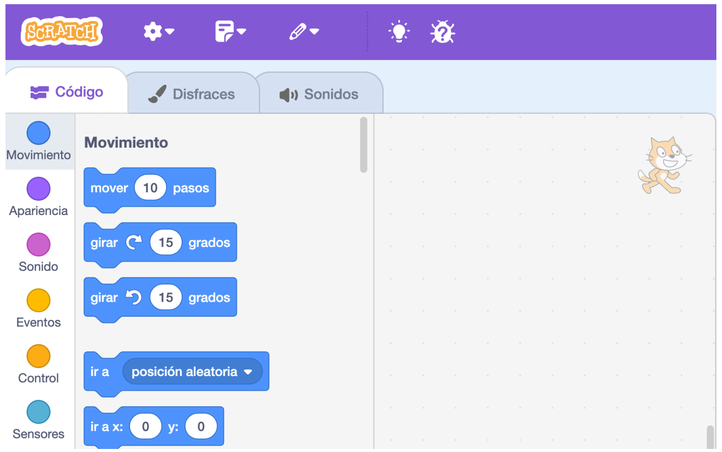
PL are often **inaccessible**, with syntax, tools, and docs that are not screen-readable, ignore neurodiversity, or require fine-motor movements, like pointing.

Carlos

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

Language

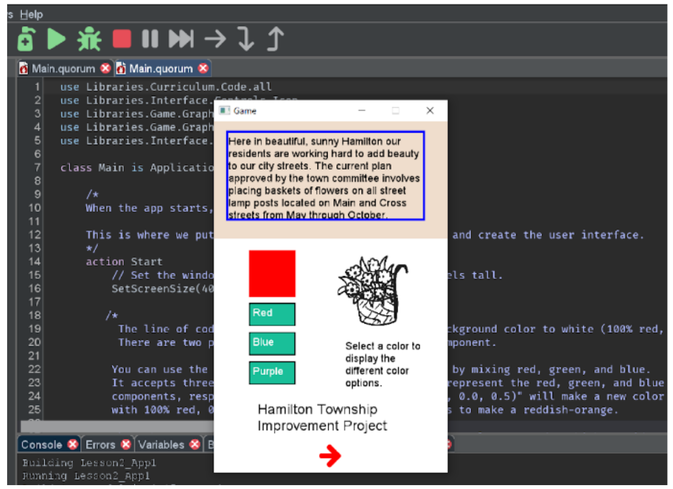
e.g., *Scratch* can be localized to non-English languages, but is inaccessible.



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

Ability

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



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

Carlos

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

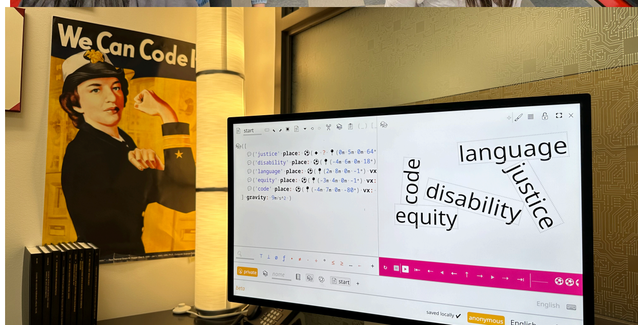
What **design**, **engineering**, and **pedagogical** challenges arise in designing at this intersection?

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

Carlos

Our approach

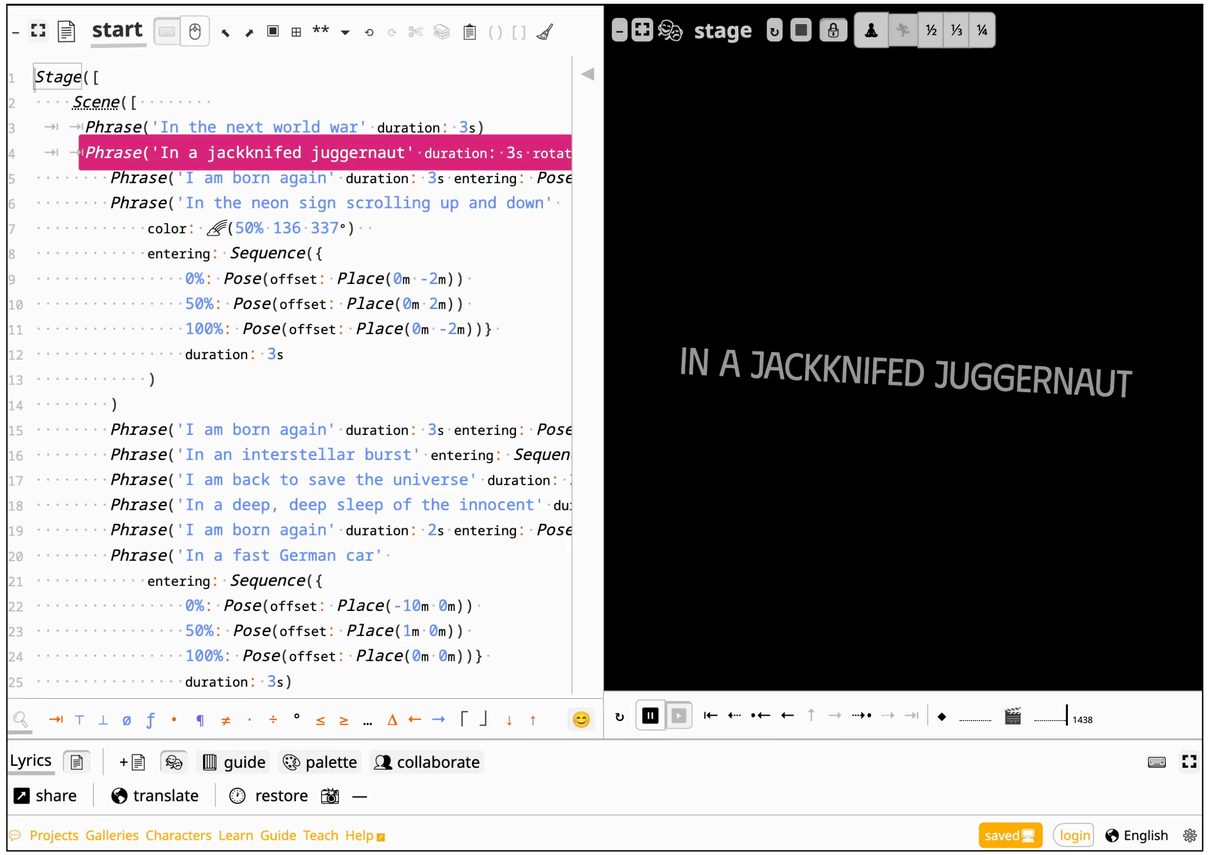
- 30 months of community-engaged PL design and development:
  - Partnering with **teachers** to understand the challenges of teaching multilingual youth and youth with disabilities.
  - Engaging **multilingual youth** and **youth with disabilities** to understand tensions with existing multilingual and 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.



Amy

Wordplay!

try it at wordplay.dev



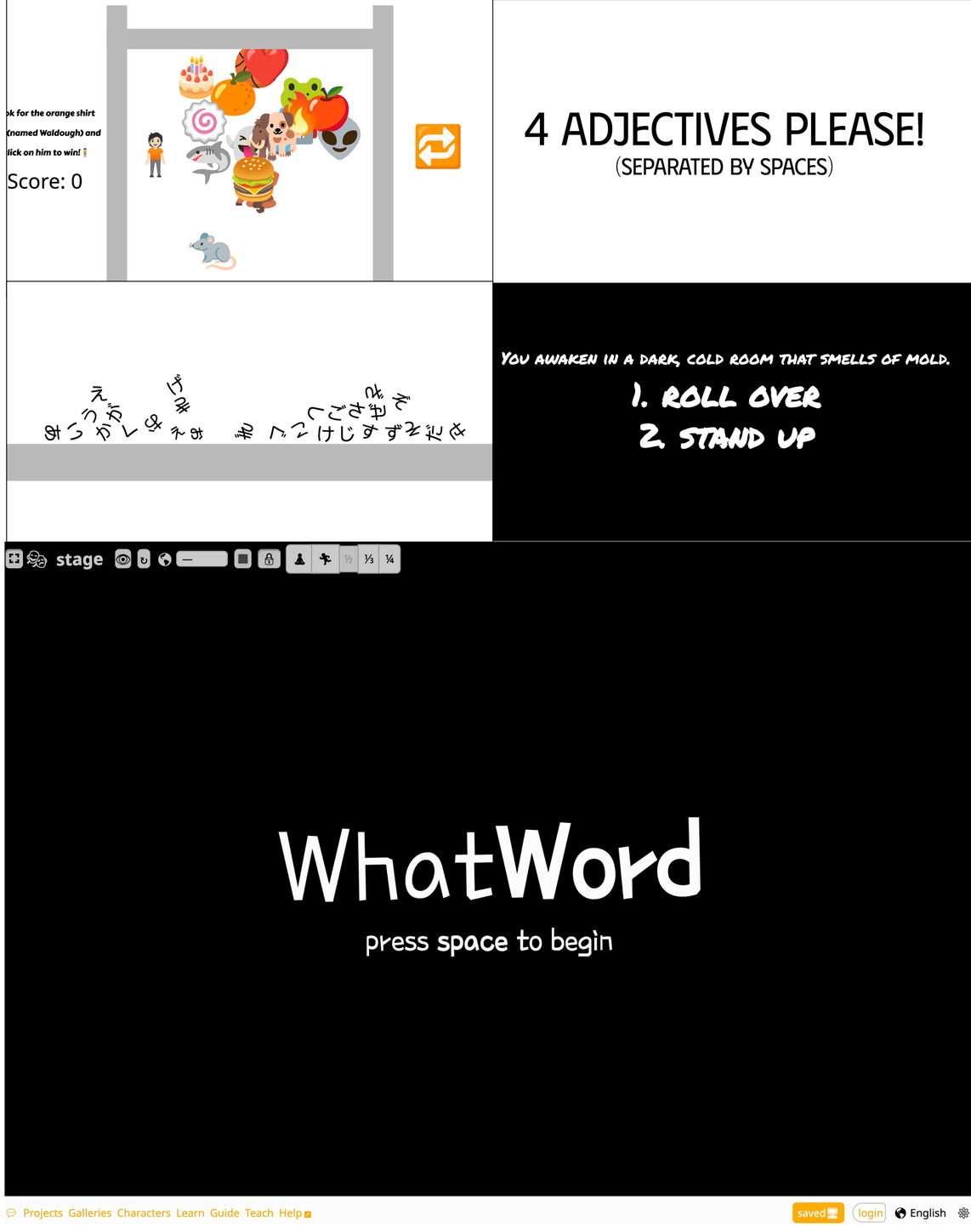
An educational programming language for creating interactive, multilingual, accessible typography.

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

Amy

1. Output is typographic

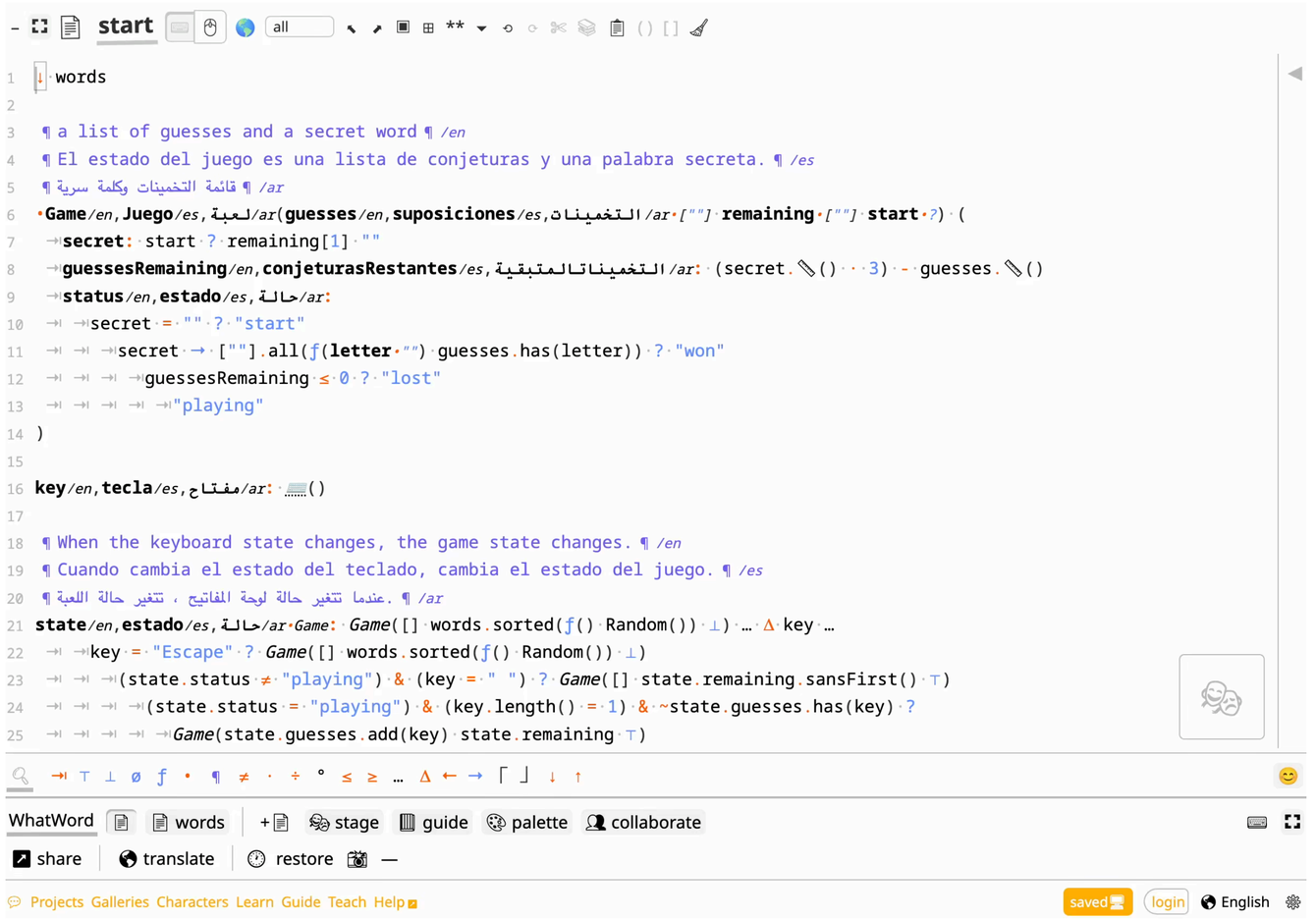
- Most educational PL use **graphics**, which are inaccessible without image descriptions, and image descriptions are usually monolingual.
- Wordplay, in contrast, uses **text**, which can be web accessible and engage culture and identity.
- Programs can generate multilingual text output, building localization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



Amy

2. Code is symbolic and multilingual

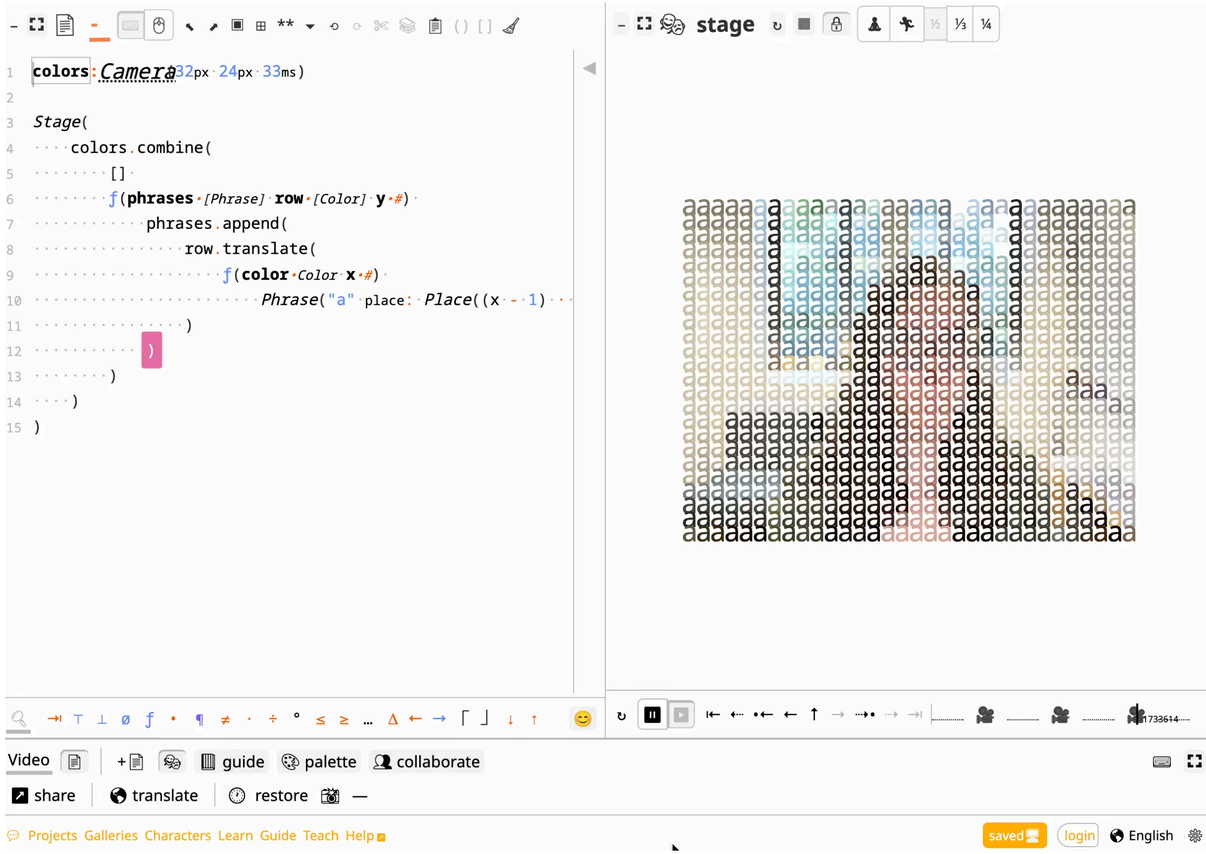
- Using **keywords** in a PL embeds a particular language in all programs.
- Wordplay only uses punctuation and multilingual text, allowing youth to embed multiple translations of **text**, **documentation**, and **identifiers** in code.
- The editor can screen read all punctuation for accessibility.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



Amy

3. Code is functional and reactive

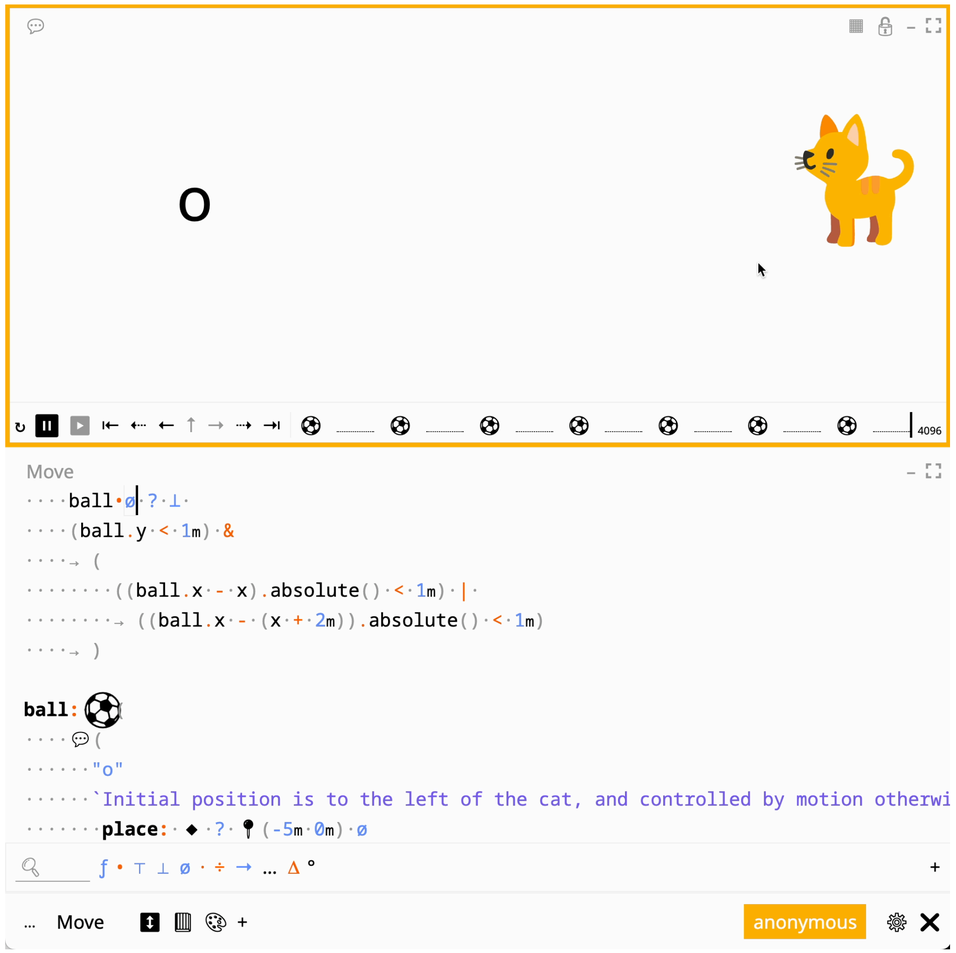
- Imperative code is verbose and does not explicitly describe its purpose, potentially creating friction in screen reading.
- Wordplay is **functional** and **reactive**, so its code is declarative, shorter, and achieves rich interactivity with minimal code, indirectly benefitting accessibility and language 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.



Amy

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 it's purely functional nature: it's only necessary to record a history of inputs, and the rest can be recomputed.



Amy

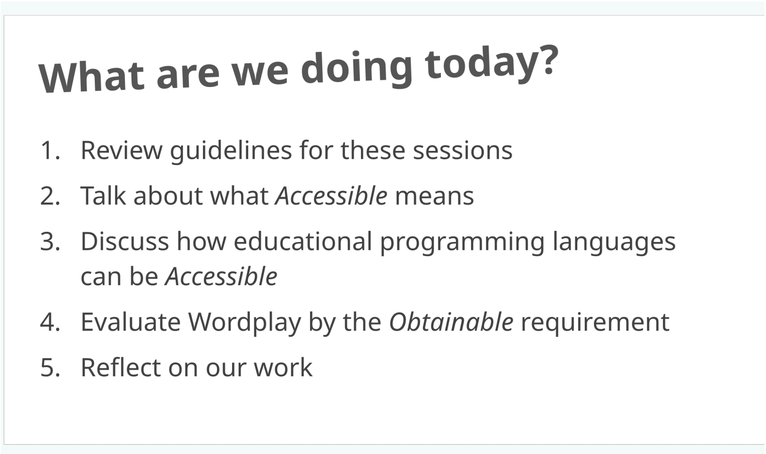
5. Governance is student and teacher led

- 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.

Amy

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.



Isabel

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.

Isabel

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.

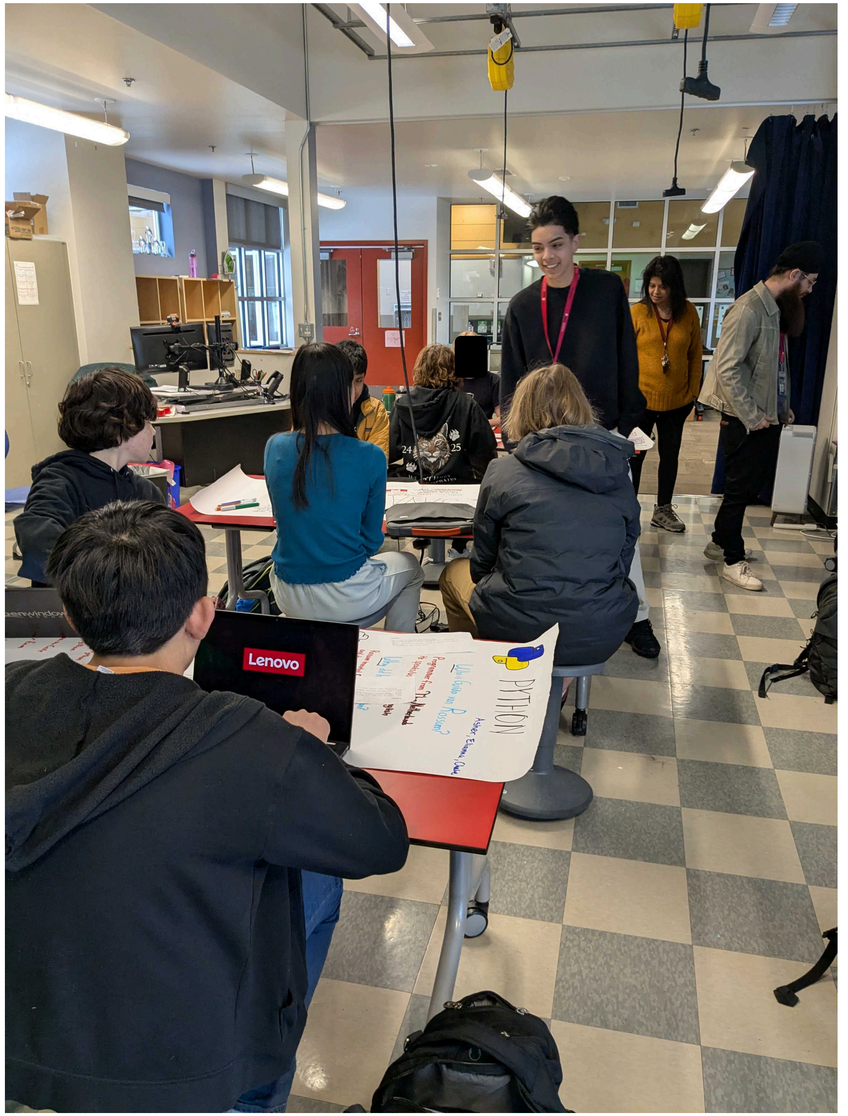
❌ It's only partially usable without internet access	❌ Youth are unsure about the benefits of functional code
❌ Smartphone layout and speed is inadequate	❌ It needs to support multilingual collaboration
❌ It needs more customizability for access needs	❌ It should frame youth as shaping future of computing
❌ 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

Adrienne

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:

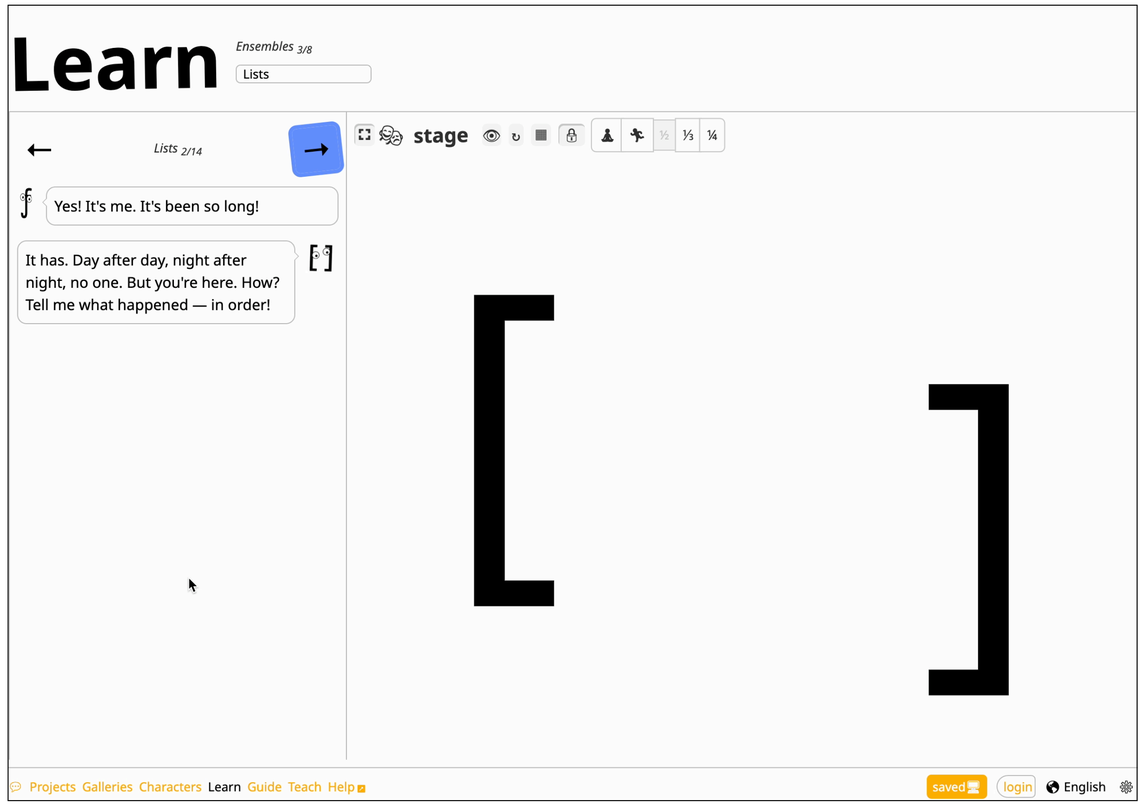
- If everything has a description for access, everything must have **thousands of descriptions** for every language and reading level.
- Focusing on text may increase accessibility and language inclusivity within the domain of interactive type, but at the expense of **other kinds of culture and identity**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



Adrienne

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.

Carlos