ABSTRACT
This research project examines esoteric, or “weird” programming languages—a class of languages not created for any “real world” use but instead for play, competition, proof of concept, or art. We have developed strategies for representing how a wide variety of these programming languages work, circulate, and make meaning for a non-specialist, non-technical audience. Through a multidisciplinary approach in which we gathered scholarship from the fields of writing studies, digital rhetoric, computer science, and graphic design, we have found three principles that apply to visualizing code, which emphasize the effectiveness of: (1) technical communication, (2) tables for stack-based languages, and (3) function over form. We then consider how our findings can be used to help others understand code and become code-literate.

RESEARCH QUESTION
What strategies are useful for representing how a wide variety of esoteric programming languages work, circulate, and make meaning to a non-specialist audience? What can we learn from such visualizations?

THESIS
Esoteric programming languages can be visualized and understood by to a non-specialist audience by using techniques from technical communication and by focusing on function, rather than form.

RESULTS & CONCLUSIONS
Visualizing creative code is helpful for understanding programming languages, particularly for a non-specialist audience, because it reveals how the code functions outside of its form as a low-level programming language. Our research found three principles that are useful to consider when visualizing code:

1. Strategies from technical communication are consistently effective in demonstrating the function of code. The most useful visualizations include Sequence Diagrams and Activity Diagrams.
2. Coding languages can be understood through function via “commands.” This approach only considers the form of the programming language.
3. Tables that contained visual cues of a sequence were most effective in visualizing stack-based languages.

FUTURE IMPLICATIONS
These findings reveal that there are:
1. Intersections between the principles and elements of design and coding
2. Non-traditional means available to allow a vast majority of people to understand programming languages

As a result, through visualizing code, we can consider what it means to be code-literate and how the definition of literacy is changing to adapt to the presence of coding.