eCodeOrama Proposal

Zhengfei Li (Alex) zhengfeili.alex@gmail.com /nick zli_alex

1 Biography

1.1 Basic Details

- Full name: Zhengfei Li
- Email: zhengfeili.alex@gmail.com
- Github Username: zli-alex
- IRC Nickname: zli_alex
- First Language: English
- Location: Philadelphia, Pennsylvania, US; Timezone: Eastern Standard Time (UTC-5:00)
- I contributed to Dive Into Systems: Exercises
  - Github Repo: `https://github.com/Dive-into-Systems`
  - Live webpage:
    - Cache Table Questions I contributed to: `https://diveintosystems.org/exercises/section-cachedemo.html`
    - I am listed as a student developer: `https://diveintosystems.org/exercises/frontmatter-3.html`

1.2 Motivations

Greetings! I'm Alex, an incoming Carnegie Mellon University M.S. in Computer Science student, passionate about educational technology.

I am passionate about engaging students of all ages with accessible and interactive educational technology. Google Summer of Code is a great opportunity for me to work on an open source project with lasting impacts. My experience building a webpage with interactive exercises for computer system courses equips me with practical skills in GUI development and user-oriented design. Therefore, eCodeOrama is perfect for me to further improve these skills while contributing to a EdTech project.

I am drawn to GFOSS ecosystem because of your mission of promoting open standard, free software, and open content, deeply echoing with my beliefs. In addition, your dedication to creating educational tools and fostering community-driven projects resonates with my experience in open source development. I am eager to contribute to and learn in such an inspiring environment with collaborative nature and active support from leading academic and public institutions in GFOSS.

eCodeOrame, in particular, combines my enthusiasm for interactive educational tools and my background in building user-friendly web interfaces. My work on an open source project for developing a dynamic exercise generator has shown me firsthand the impact of intuitive GUI design in enhancing learning experiences. I hope to apply these skills to a project that uses Scratch and interactive code teaching, helping to make learning to code both fun and accessible.

During the program, I hope to receive valuable mentorship and feedback that will help me grow in my technical skills as well as my understanding of the EdTech industry. I also expect a collaborative environment of problem solving where I can contribute to and learn from a community of developers. After successful completion, I ain to maintain a continued relationship with GFOSS, hoping to collaborate and contribute to future projects.

2 Project Information

2.1 eCodeOrama: Project Overview

eCodeOrama is an interactive tool designed to automatically extract, visualize, and allow editing of the event-based flow in MIT Scratch programs. By parsing Scratch project files (which are stored as JSON within a ZIP), the tool will extract key entities—such as sprites, scripts, events, and message broadcasts—and arrange them in a clear, CodeOrama-compatible layout. The layout is essentially a table where each column represents a sprite and each row corresponds to an event (for example, a broadcast message or a user interface trigger). Arrows will be drawn between cells to indicate how events trigger scripts, which is particularly useful in visualizing concurrent threads and message-based communication in Scratch.

The project can have a lasting impact on GFOSS:

- Educational Enhancement: The tool simplifies the understanding of complex Scratch projects by presenting a high-level visual flow, aiding students and educators in debugging and code explanation.
- Debugging Aid: By clearly outlining which scripts trigger which events (and vice versa), the tool will help uncover issues such as unconnected broadcasts or unreachable code segments.
- Community Contribution: Being open source, eCodeOrama invites contributions from the educational community and other developers, promoting further enhancements and adaptations to different block-based programming environments.

2.1.1 Technical Tools

Programming Languages:

- Python: For parsing JSON, backend processing, and possibly a desktop GUI (using PyQt or similar).
- JavaScript: For an interactive web-based visualization interface (using libraries like D3.js or scratchblocks for authentic block rendering).

Key Libraries and Tools:

- Parsing: Python's built-in JSON libraries or dedicated libraries like sb3 for Scratch file extraction.
- Graph Representation: NetworkX to manage the flow graph of sprites, scripts, and events.
- Graphical Layout: Graphviz (using DOT language) for initial layout and edge routing, with enhancements for interactive adjustments.
- User Interface: PyQt (for a desktop application) or modern web frameworks to create a rich, interactive experience.
- Export Options: Ability to export layouts to PDF, JSON, XML, or spreadsheet formats (XLSX, ODS).

Expected Results:

- A fully functional interactive tool that automatically visualizes the event-based flow of Scratch programs.

- An intuitive interface where users can customize the layout (e.g., reorder sprites and events, fold/unfold code nodes, adjust edge routing).
- Multiple export formats for printing or further processing in external tools.
- Detailed documentation and user guides, ensuring that educators and students can readily adopt and benefit from the tool.

2.2 Background Understanding and Repository Research

In an effort to prepare for the development, I performed a review and research on the project repository and associated supporting documents. A summary of my understanding is provided below:

2.2.1 codevisionG.pdf:

Although its domain appears to address an automated agricultural control system (with elements such as moisture sensors, motor activation for irrigation, and fertilizer control), the underlying concept is analogous to what eCodeOrama aims to achieve. It demonstrates a step-by-step process for transforming raw sensor data into a coherent, interactive visualization. This "code vision" approach reinforces the importance of clear, automated, and configurable visualization in complex systems—a principle that will be applied to the visualization of Scratch's event-driven code flow.

2.2.2 Ideas.html:

This document lays out the technical blueprint for the project. It explains that a Scratch program is composed of multiple sprites and scripts, with each script starting with an event block. The document advocates for creating a graph-based model where scripts are nodes and the broadcast messages form directed edges. It also discusses:

- Various output formats, including text-based reports and graphical layouts.
- Interactive features such as reordering columns, folding/unfolding nodes, and customizable edge layouts.
- Challenges such as uneven column lengths and space-saving techniques.

These ideas form the core functionality and interactive aspects that eCodeOrama must provide.

2.2.3 TassosLadias.html

This file, authored by Tassos Ladias, provides historical context and validation for the CodeOrama concept. It explains that CodeOrama was conceived to meet a real need in the educational sector—helping educators visualize Scratch programs. The document highlights:

- The research behind CodeOrama, including usability studies (e.g., using the SUS method).
- Adoption examples, such as its use in educational robotics competitions and inclusion in academic publications. This background underscores the proven educational value of visualizing code flows and justifies the necessity for an automated, interactive tool like eCodeOrama.

2.2.4 README.md

The README file summarizes the overall project goals and expected deliverables. It outlines:

- The need for a tool that can parse Scratch projects and visualize their event-based execution flow.
- Technical requirements such as parsing JSON, building a two-dimensional layout, and supporting interactive configuration.
- A phased implementation plan with clear milestones and deliverables.

This document serves as the central project overview, aligning the development efforts with educational objectives and community needs.

Collectively, these documents demonstrate a comprehensive understanding of both the conceptual underpinnings and practical challenges of building eCodeOrama. They show that the project is deeply rooted in both research (with references to studies and prior implementations) and practical implementation (through detailed ideas on interactive layout, export formats, and usability considerations).

2.3 Project Schedule and Deliverables

Below is a detailed 12-week timeline outlining key milestones, deliverables, and buffer periods. This plan is realistic and designed with the flexibility required by Hofstadter's Law.

2.3.1 Week 1-2: Setup and Parsing

- Set up the development environment (Python, necessary libraries, frameworks).
- Review the repository and supporting documents (Ideas.html, TassosLadias.html, README.md, codevisionG.pdf) to refine the project scope.
- Define data structures for parsing Scratch projects (sprites, scripts, events, and broadcasts).
- Implement a parser for Scratch 3 projects (.sb3 files) that extracts sprites, scripts, and key events.
- Generate simple text-based reports showing which sprites broadcast and receive specific messages.
- Test the parser with sample projects to ensure robustness.

2.3.2 Week 3-4: Prototype and Visualization

- Develop a static graphical prototype (using Graphviz or a basic HTML/SVG setup) that displays sprites as columns and events as rows.
- Map the parsed data onto a grid layout to form a preliminary CodeOrama.
- Enhance the visualization to include actual code blocks rendered via scratchblocks (or a similar tool) for authenticity.
- Build an initial interactive UI (desktop using PyQt or a web page) with basic navigation (zoom, scroll).

2.3.3 Week 5-6: Interaction and Mid-term release

- Add interactive features: allow users to reorder columns (sprites) and rows (events), and fold/unfold individual script nodes.
- Implement save/load functionality for user-customized layouts.
- Conduct extensive testing and bug fixes.
- Prepare a stable mid-term version (Version 1.0 Alpha) that demonstrates core functionalities.
- Prepare a mid-term report with screenshots and a demonstration of key features.

2.3.4 Week 7-8: Debug aids and visual refinements

- Integrate debugging features (e.g., flag unmatched broadcasts, highlight connections).
- Incorporate support for displaying Scratch comments as presentation directives.
- Refine edge routing and visual styling (e.g., color-coding of arrows, improved layout rules).
- Enhance UI responsiveness (adaptive layouts, scrollbars, zoom features).

2.3.5 Week 9-10: Debug aids and visual refinements

- Add support for Scratch 2 projects (.sb2) by adapting the parser.
- Implement handling for user-defined blocks (procedures) and refine clone event management.
- Create comprehensive documentation and a user guide (with clear instructions and examples).
- Conduct user testing and refine the interface based on feedback.
- Prepare additional materials (e.g., demo video) for the final evaluation.

2.3.6 Week 11-12: Wrapup for submission

- Use buffer time to polish the application and incorporate any optional features (e.g., execution trace simulation).
- Finalize export options (PDF, JSON, image, etc.).
- Perform thorough final testing across platforms.
- Prepare the final project report, demonstration materials, and package the application.
- Discuss future work and long-term maintenance with mentors before submission.