Application for Interactive LED Visualization
University of California, San Diego Computer Science and Engineering
Hayden Gomes

Abstract – The University of California, San Diego’s Department of Computer Science and Engineering has over 200 industrial tri-color LED fixtures that are significantly underutilized. This Android application works to increase the LEDs’ accessibility by providing a simple interface for programming visualizations. The web server acts as an easy to use and widely accessible interface to the LED hardware.

I. Introduction

Industrial tri-color LED fixtures have a wide range of applications from stage lighting in theaters to accent lighting in stores, museums, and clubs. There are a variety of professional systems for controlling these displays with prices to match. The University of California, San Diego’s (UCSD) Computer Science and Engineering Department (CSE) has a 4 x 70 array of fixtures with a simple control system that provides little customizability in visualization programs.

In an attempt to develop an alternative solution for controlling the LEDs, Catherine Wah, Emmett McQuinn, and Hayden Gomes developed a visualization language. The language compiles into an array of colors that are then transmitted through a USB adapter to the LEDs.

In this paper I will present the continuation of this project. With the goal of increasing the accessibility and mindshare of department’s LED fixtures, I developed an Android OS application. The application allows users to program in the visualization language and submit their code to a web server for compiling and execution. This application is being developed for a smart phone and not a pc because of the inherent mobility and on hand nature of cell phones. The expectation is that students will be more apt to use the application if they do not have to plan ahead and have a laptop on hand when they want to control the lights.

II. Background

The University of California, San Diego’s (UCSD) Computer Science and Engineering (CSE) department building was constructed in 2005. The architecture of this state-of-the-art teaching and research facility was designed with a unique look and feel. Along these lines the floor plan of the building is shaped like a foot. In the “big toe” of the building tri-color LED fixtures were installed. With seventy fixtures lining the large windows on each floor of the big toe, these LEDs are able to fill the room with colorful ambient light.

II.A LED Hardware

The lights are iColor Cove fixtures. This is the first model in a successful line by Color Kinetics. The fixtures work in a parallel circuit with one data line. Using DMX-512 signaling protocol, a standard for theater lighting equipment, each fixture is individually addressable to receive 3 channels for RGB intensities. Through this convention, each fixture can be independently set to one of 16.7 million additive RGB colors. The fixtures are capable of a refresh rate of up to 40Hz [1].

Color Kinetics Multi Synchronizer panels were also installed on each floor of the big toe. They are capable of displaying eight preprogrammed visualizations on the LEDs in the same room.
II.B CSE 231 Project

During the fall of 2009 Catherine Wah, Emmett McQuinn, and Hayden Gomes worked on a project for a course on compilers (CSE 231). The project goals were to develop a simple language for describing visualizations, a compiler for the language, a simulator, and a means for displaying the visualization on the LED fixtures. The project’s code was developed in Python. The visualization language is compiled into a 4 dimensional array of RGB color intensities of each LED fixture for every time step.

The team was successful in completing the project goals. The language has vastly increased the control and customizability of visualizations for the LEDs. However the process of developing this language revealed several limitations that needed to be addressed; the most significant of which being accessibility. The big toe rooms include offices and a conference room, all of which require a key or card swipe access. Also, the team connected directly to the LEDs using an Enttec DMX USB Pro which cannot be networked. This means that to execute the system, the computer must be hooked up in the big toe.

III. Android Application

The application was developed on a HTC Dream (T-Mobile G1) phone. While the latest version of the Android OS is 2.1, this model only supports the Android 1.6 platform. However there is sufficient backwards compatibility between versions that the application should function on all Android phones.

The application provides a simple user interface (UI) for programming in the visualization language. A database allows users to store their programs. Through interactions with the web server, the application is able to produce a simulation of the programs or have them displayed on the LED fixtures.

III.A User Interface

The UI takes advantage of Android’s views to provide a clean and intuitive interface. The application is divided into three tabs: the code editor, list of programs, and simulator. The code editor, as shown in figure 1, provides a basic text area for writing code.

![Figure 1: Application code editor UI](image)

In addition to the code, the editor allows users to provide a name and brief description of the visualization produced by the program. A popup provides users with menu options for compiling the program or to save any changes.

III.B SQLite Database

For storing programs there was a question of whether to host a centralized database on the web server or for each user to have an independent database on their phone. I chose to have the database on the Android because it doesn’t require a secure method of associating programs with their
authors and so that the phone doesn’t have to communicate with the server for the user to manage their programs.

The Android OS has built in support for a SQLite database. The application’s database uses one table with fields for the program’s ID, name, code, description, and modified timestamp. The user interacts with the database through the Programs tab as shown in figure 2. Using Android’s ListView, this tab lists all programs along with their descriptions and modification timestamps.

III.C Simulator

The simulator’s design is similar to a slideshow program. The web server hosts png images of each frame of the simulation. The application uses Android’s WebView to interact with JavaScript code. Figure 3 shows the application’s simulator tab.

By selecting one of the programs in the list, a context menu pops up with options to load the program into the editor or delete it from the database. The database is pre-populated with example programs. These programs provide users with a means to interact with the simulator and LEDs before they start to write their own code.

IV. Web Server

When a user chooses to compile their program, the code is uploaded to the server where it can be compiled and executed. Since the majority of Android phones owners have data plans, and the phones also have built in WiFi, an internet based submission process seemed ideal. The application posts the program code to a website form hosted on an apache web server. From there the website’s back end compiles the code and turns it into a simulation or displays it on the LEDs.

IV.A Web Form

As the original code from the compilers project was all written in python, the website was developed in Python with CGI. Because the form is an actual website, any internet capable device will be able to submit programs. This extends the LEDs accessibility to anyone in the department.
instead of just those who have access to an Android phone.

IV.B Simulation and LEDs

After code is posted to the site it is passed into the original compiler from the CSE 231 project. The 4-dimensional array output is then passed into a program that uses the Python Imaging Library to generate the frames for simulation.

A modified version of the original LED interaction software takes the array and transmits it through a serial connection to the LEDs. The program cycles three times and then disconnects the USB connection so that the next submission can be displayed.

V. Results

The application is near completion. While it was successful in posting to a test web form there were some difficulties getting it to properly interact with the final website. Some additional testing on a temporary apache web server is required before allowing the application to submit code to be displayed on the LEDs.

The web server is running and successfully compiles programs and simulates the output. The LED connection has also been tested and the server can also display compiled programs on the fixtures. For security purposes, additional work and testing will be required before the LED fixtures will permanently be accessible from the website.

VI. Conclusion and Future Work

The project was a success in helping make the LEDs more accessible, although it will be a while before I can tell if the usage has increased. There still remains a lot of work to be done on this project. The most important task for getting people in the department to start using the LEDs is finishing off the website. This will require some form of security be added so that the LEDs are only used by people within the department. Also, because the big toe is host to offices and a conference room, it is important that the use of the LEDs is restricted to appropriate times and some override shutoff ability is included.

VI. Acknowledgements

This project was for CSE 237D under the guidance of Professor Ryan Kastner. Dave Wargo, the CSE facilities, lab, and equipment technician aided in the connecting of the LED hardware to the web server. Chris Kanich and Mikhail Afanasyev provided the web server and assisted with configuring the website. My original team members, Catherine Wah and Emmett McQuinn continued to work on the project and helped get website and simulator working.

Bibliography