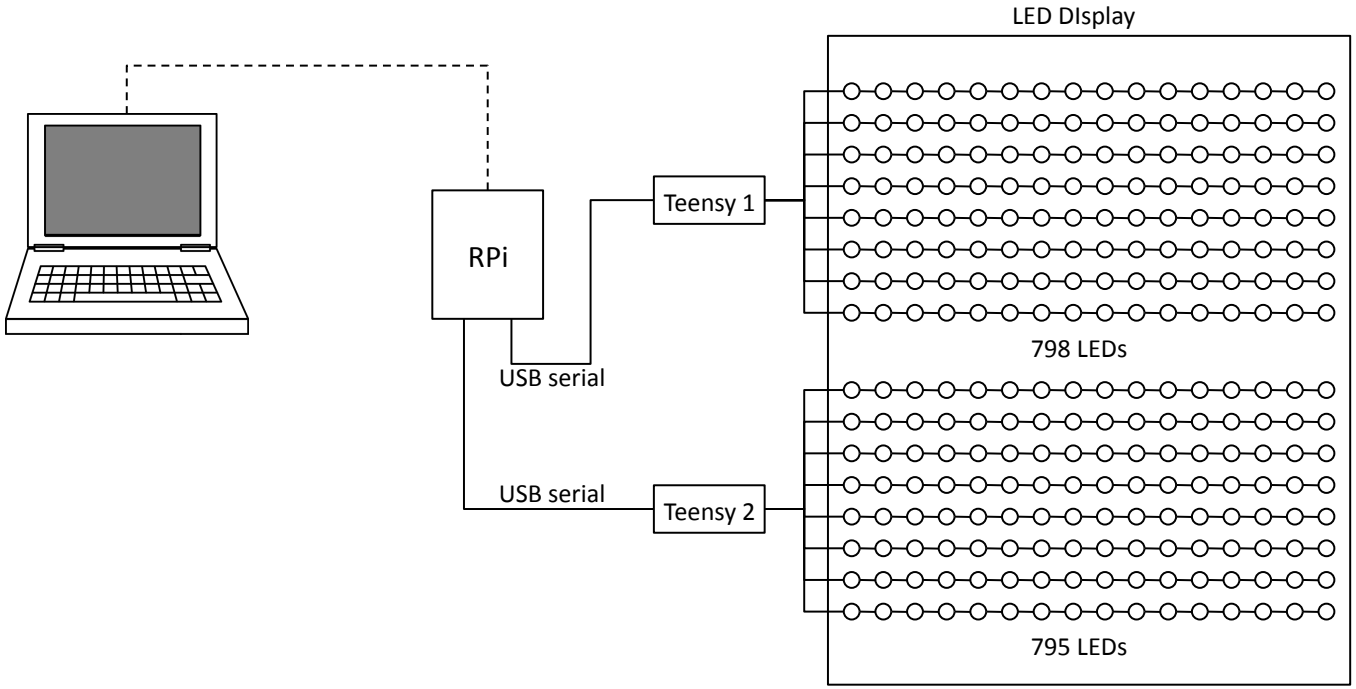
Irregular LED Display Project

Image Processing Algorithm Design

Bill Tubbs

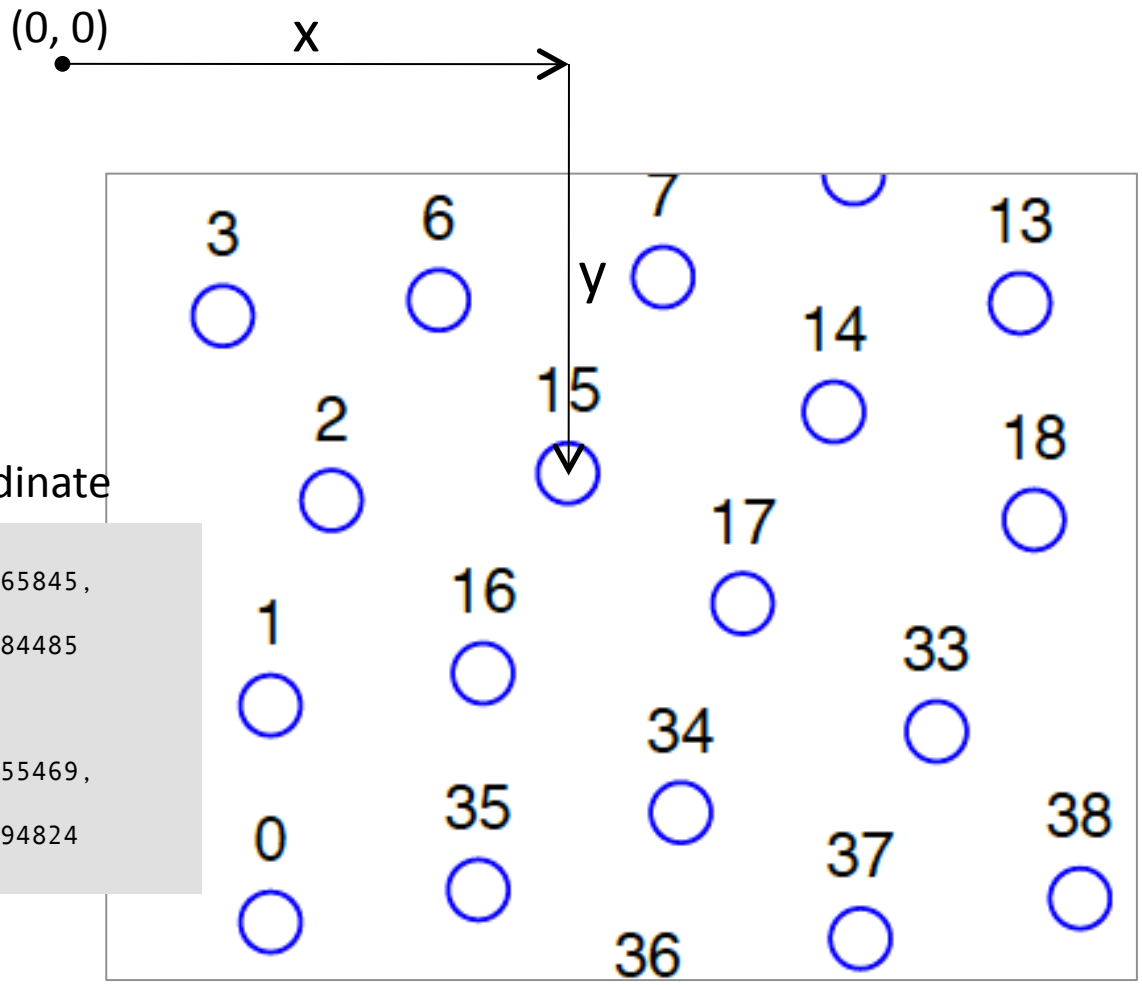
November 2015

Hardware



- Raspberry Pi 2 Model B, 900MHz quad-core ARM Cortex-A7, 1GB RAM, with Raspian OS, and Python 2.7 installed
- Two Teensy 3.1 Microcontrollers, 72 MHz 32 bit ARM processor, 65kB RAM, 262kB Flash memory, running Arduino C with Teensyduino and OctoWS2811 LED Library

Irregular LED Arrangement



Each LED has an (x, y) co-ordinate

```
const float centres_x[] = {
    16.0602016, 16.0396748, 30.3565845,
    ...
    1891.67322, 1917.64624, 1963.84485
};

const float centres_y[] = {
    464.544983, 413.822815, 365.855469,
    ...
    1990.73901, 1948.36438, 1973.94824
};
```

Supersampling

For antialiasing the easiest solution is supersampling: rasterize on a higher resolution grid and then downsample to your irregular grid. The higher resolution grid could be an irregular one or a regular one. Again considering the low resolution, simply rasterizing a standard rectangular image, then blurring it and selecting a fixed set of pixels at your pixel positions, could be an easy way to live off the existing graphics pipeline.

Steve Marschner <srm@cs.cornell.edu>

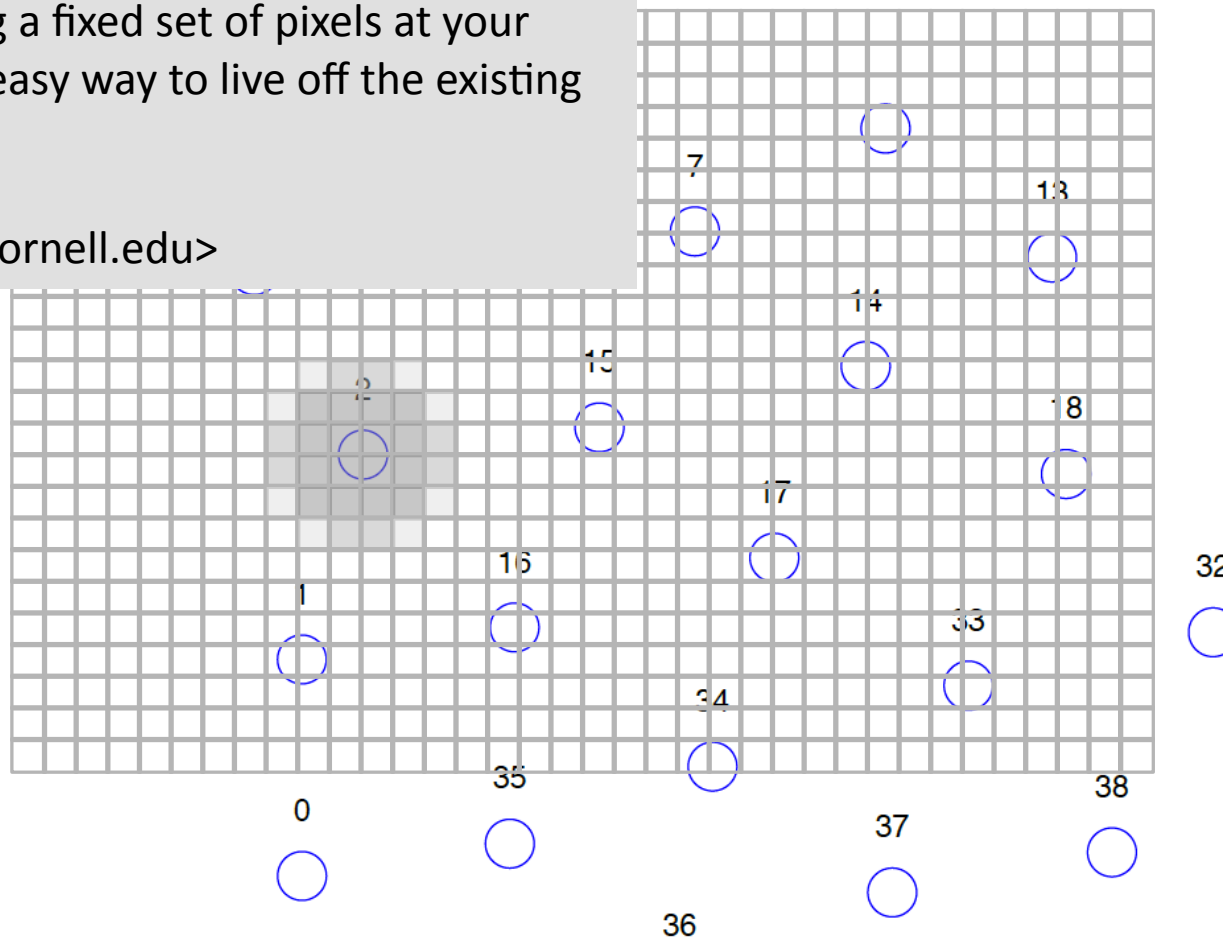
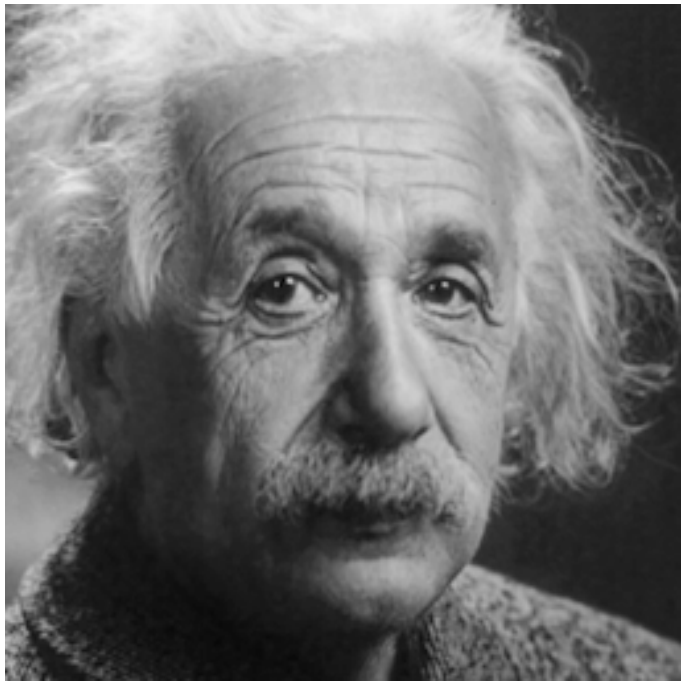


Image Mapping



256 x 256 pixels

1593 pixels

Graphics Pipeline