Software Design Overview

# Terms and Concepts

First we will implement a basic system based on a generally useful mathematical model.

When the basic model is proven, we will try to generalise the software to allow for much richer lightshows. Allowing for “virtual” pixel strips will allow lightshows to work at different scales, and to work in spatial directions that are not constrained by the hardware strips. Allowing for lightshows made of lightshows will enable users to build up longer more complex patterns with minimal effort.

## Basic Entities

1. **Pixel Strip**

An actual physical NeoPixels LED strip.

1. **Lightshow**

A cyclical pattern of lights that changes over time for a certain period of time. A lightshow is defined in terms of a tables of colours, a length of time, and other parametres. Each lightshow also is assigned an algorithm to control how the defined colours are combined. In general, a list of colours is either treated as a sequence in time or a sequence in space. A lightshow can be applied to a Pixel Strip.

1. **Master Light Show**

A special lightshow that is composed of a list of other lightshows.

1. **Active Light Show**

A lightshow that is currently showing.

## Proposed Advanced Entities

1. **Physical Pixel Strip**

An actual physical NeoPixels LED strip.

1. **Virtual Pixel**

Can be either an individual LED in a Physical Pixel Strip, or an entire Physical Pixel Strip.

1. **Virtual Pixel Strip**

An ordered set of Virtual Pixels. Can be treated like a Physical Pixel Strip.

1. **Primitive Lightshow**

A lightshow that is not composed of other lightshows. Only primitive lightshows can have a table of defined colours. In addition to lists of colours, there can be matrices of colours. A matrix of colours is treated as sequences in space appearing sequentially in time.

1. **Symbolic Lightshow**

A lightshow that is defined using the same notation as a primitive lightshow, but is allowed to combines other lightshows instead of combining colours.

1. **Master Light Show**

A special lightshow that is the top node of a hierarchy of lightshows.

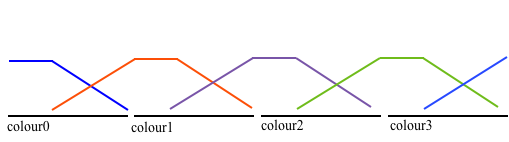
# Basic Lightshow Concepts

Here are the basic concepts underlying the Electronic Moon software design.

## Lightshows have cyclical sequences of colours

Primitive lightshows are defined by providing a simple list of colours. The list will be treated as an evenly divided sequence. For example, consider a lightshow defined with the following list:

blue, orange, purple, green



Because there are four colours in the list, each will be spread across one quarter of the pixels in a strip. If the colours are meant to blend together smoothly, then each of the four colours will have the same number of full intensity pixels and the same number of pixels with blended colours.

If the lightshow did have smooth blending, then there would be a region where the green blended back to blue, in order to complete the cycle.

## Lightshows exist in time and space

Every lightshow is assumed to affect all the pixels in whichever pixel strip it is applied to. It is also defined to last a specific length of time. Lightshows will allow users to make patterns in either space or time.

There will be at least two types of basis functions used in Electronic Moon. A “spread” function distributes a sequence of colour in “space” (i.e. all colours will be visible at once) and will remain unchanged for the duration of the light show.

A “fade” function distributes a sequence of colour in time, and will colour all the pixels in a pixels strip that same.

The the example above related to a “spread” function, but the same sequence of colours could also be used to illustrate a fade. In other words, given a sequence of blue, orange, purple, green, all the pixels would be blue, then fade to orange, then fade to green, then fade to blue again.

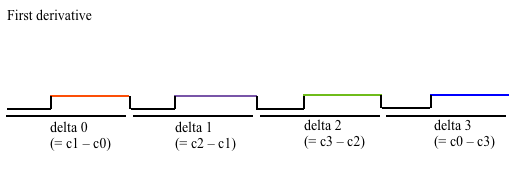
## LED values are never reset

The basic loop algorithm allows for each LED to have its value adjusted every fraction of a second. In order to allow for efficient multiple overlapping lightshows, the algorithm assumes that each lightshow will only add (or subtract) a value for each pixel, never reset a pixel value. If any lightshow did this, it would interfere with other overlapping light shows.

Lightshows achieve a fade by incrementally adding a colour until the full colour has been reached. Spreads are achieved by setting the colour of all pixels and then leaving them unchanged for the rest of the lightshow. This only works because we assume that LED values are never reset.

## Colour values are coverted to deltas

The “delta” values are the actual numerical values that are added to LED values during a single iteration of the basic loop. It is the delta values that are stored in program memory. The delta can be thought of as the first derivative of the lightshow.



The concept applies in either space or time. If we visualise the first derivative of the lightshow, we can see that linear fades or blends in a lightshow can only result in flat pulses in the first derivate. Pulses will range proportionally from long and low, to tall and spiky. This gives us a single “smoothness” parameter that users can use for lightshows in general (we only plan to have linear patterns).

## Applying deltas in time and space

Fade lightshows could require a different algorithm from spread lightshows, or any other type we might dream up. Therefore the algorithm for applying a delta must be built into a function and the choice of function is one of the parameters for a lightshow. We call such a function a “basis function”, because if it is designed correctly the function can handle any number of colours in a sequence.

Having said that, it is possible to actually come up with a general fade/spread basis function if we imagine that fades and spreads are both examples of a general colour delta matrix.

delta\_x0

delta\_x1

delta\_x2

delta\_x3

delta\_t0

delta\_t1

delta\_t2

delta\_t3

In this view, a spread can be thought of as a row matrix with “x” deltas. A fade can be seen as a column matrix with “t” deltas. As long as we constrain all columns or all rows to be the same size, we can accommodate more general lightshows where we fade from one spread to another. This would result in a rectangular matrix of “x-t” deltas.

## Algorithm

## Data Structures

At load time, all Pixel Strips and Lightshows will be uploaded into program memory. The Master Light Show provides the hardcoded top level instructions for the program. These lightshows in program memory are treated as a library, to be used multiple times during the duration of the Master Light Show, and may be used on more than one strip at one time.

For each variety of lightshow, there will be a “basis” function in program memory that is capable of processing the sequence of colours in time or space.

At run time, a dynamic list of currently active lightshows is created. We call this the APL list (for Active Primative Lightshow). Each entry in this list represents a single instance of one of the lightshows defined in the library.

A list will also be created to contain the current colour values for all the active physical strips (APS).

## Basic Loop Structure

The software mainly consists of a simple loop, repeated over and over (our target rate is 16 loops per second). Here is the algorithm.

for each active primitive lightshow (APL) in the APL list

use the basis function pointer with parametres, strip ID, current frame

if the APL is finished

terminate the APL

(the APL must have already cancelled its last colour)

send its address to its parent active symbolic lightshow

(the parent will pull its list out of linked list)

for each strip in the active physical strip list

show the strip