A Performance Instrument for Lighting

Louise-Philippe Deners
745 rue Guy, suite 300
Montreal, Quebec
CANADA
H3J 1T6
514.839.4327
e-mail: lucy@McGill.McGill.EDU

Abstract
@FL is an interactive graphical programming environment based on the object oriented paradigm. Lighting cues are not only seen as keyframes but more as a set of behaviors/processes that interact to achieve the final look on the stage. Traced and shared control creates a performance instrument for lighting. Current implementation runs on Macintosh II and supports MIDI and DMX512 dimmer protocol.

The author is grateful to the Canada Arts Council, Media Arts Section, which generously supported this research. The author also wishes to thank the Banff Centre for the Arts, Media Arts Section, for providing creative residencies and Philippe Jean for implementing major components of the system.

1. Motivations.

Control, rehearse & performances.

The usual technique which consists of building cues is a functioning solution. The control of lights is described at its final stage, as a series of numbers indicating the intensity (or parameter values) of each instrument (DMX512). A cue is no more than a simple vector of numbers and it is similar to a keyframe caption in a story board. Interventions are created by crossfading from one cue to another or of breaking them into smaller parts. We must take note that a crossfade goes through a fair amount of lighting levels even though two simple keyframes were given.

Rehearsal process demands rapid change and somewhat immediate response to the director. A cue is of low complexity and it is manipulated with simple and real-time efficient editors. However, a cue has a static and rigid description while the show is running.

Live accompaniment brings along a myriad of faders and buttons which after proper training become an extension of the human operator. Computerized consoles currently act as sophisticated memories where light cues and light groups (e.g. submasters) are memorized throughout the rehearsal and played back (retriggered) live by a human operator.

Current MIDI techniques.

MIDI was recently introduced into the lighting consoles. The protocol consists only of mimicking the operator's actions to achieve remote control and accurate synchronization. MIDI accesses all faders and buttons. Current techniques are using a lot of sequencing, thus bringong along problems of a rigid framework. It is hard to change a level and assure light balance throughout the show.

Tradeoffs such as sequencing the light washes and fine tuning cues on top are usually suggested. MIDI refraining small sections of operations along with human interventions appears to be a workable combination of both worlds. It is quite obvious that usual MIDI devices are not the perfect tools for live lighting operations and even screening of performances.

Complexity.

The amount of design and performance parameters (stage sensors, scripts, the musical gesture, etc.) inhibits the exhaustive
description of specific lighting cues. Furthermore, sophisticated control schemes have to deal with the on stage creative process: alterations, incremental refinement and rollback/goto discrepancies.

Looking at music visualization, one would be interested in creating a light pattern over performers depending on their respective monitored pitch, density and intensity. One would like to change color when the computer detects a certain condition. The performance becomes a duet between the musician and the lighting designer. This simple scheme encompasses the complexity of the usual lighting cue.

Alternate control schemes.

@FL suggests algorithmic cues or behaviors as descriptions of lighting activities. For example a cue that bursts fading lights could be described as:

"flash @ 20 % all lights > 50% when also fading out during the last 20 seconds"

We observe that this cue is contextual in time, hard to recreate by keyframes and cumbersome to track by a human operator. The cue is also described independently of any other cues that are simultaneously running. All ingredients to move away from a rigid framework.

You can regard @FL as a musical synthesizer where you turn a series of knobs until you create the desired sound. Likewise, let the computer do the clerical work, eg fades, and we decide on the important parameters: the machine monitors the speed and density of musical performers and we control the colors. Here is a list of simple behaviors:

"A happens quickly when B and C occur simultaneously, otherwise A happens slowly"

"If a performer is in an area, light level > Min level otherwise the light < Max level"

"When a special is brought up, all instruments focused in this area are lowered by 20%"

"A follows the tempo of a performer"

2. @FL: a testbed.

Integration of software components along with various external devices, e.g. MIDI, is a strong prerequisite in such a platform. Object-oriented environments such as MAX, preFORM toolkit, Synchron. Forms and the Creation Station are various examples. @FL is a variation on the MAX score, customized for lighting at both interface and scheduling levels. Rules as described in the previous sections are to be implemented by programming in the environment.

Objects/behaviors construction.

@FL is an interactive graphical programming environment. An object in the system is a box that processes information. An object possesses inlets and outlets corresponding to input and output of the process. The designer is offered a set of built-in objects and a desired behavior is implemented by interconnecting objects' inlets and outlets together with links. Information flowing along the links corresponds to messages sent from object to object.

A set of interconnected boxes can be grouped into a patch that can be further used as an object. Therefore, hierarchical description is supported. Information handled by the system (messages) includes simple data types: integer, float and time (SMPTE or msec). Along with simple data types, complex types such as MIDI raw, vectors and general objects can flow on links. Vectors usually carry the equivalent of a cue and are a table with statistical and symbolic attributes.

Typical program architecture.

Patches to run a performance tend to take a similar architecture where we have an input filtering stage and dispatch, the core of the processing and final arbitration (global behavior) stage. For the processing we usually have a driver layer where its instances are used in several contexts along with a set of control data.

Consider a simple music performance where we have three performers on stage. Among all the lights, each player as a set of backlights and front specials. We are interested in

ICMC 472
having a small behavior that controls lighting for each player with the following parameters: color, warm selection and back/front levels. We then create these instances of this "performer driver layer" that take care of controlling each group. Afterwards, we can bundle these patches in several ways.

Given a set of behaviors, arbitration has to be performed before the current state of lighting is produced. It is at this point that the designer decides to give more weight to specific patches. A master level fader, for example, intervenes at this very last stage. Light balance or global behavior is implemented at the arbitration level.

Data Management.

Information gathered throughout the design process has to be organized, memorized, and retrieved. A straightforward roll-back mechanism is used via presets. Presets are embedded in each object and can be recalled to restore a specific state of an object. Presets are then created at the beginning of performance sections or major moments. Patches have to be created to broadcast preset requests to proper objects.

Bags are objects that collect incoming data stream and offer retrieval services such as: next/previous, get all, find, etc. A bag can be used to simply store cues or create a data base of predefined states. Spies tap into a link, memorize and retrieve incoming stream. Upon request, the spy plays back its information, merging or not with the incoming stream. Spies are useful when time critical tasks are computed and are too slow to play back in real-time. The signature is usually Time Code but might be defined symbols. Journals are basically spies but they are targeted at a whole set of objects without having to tap into the links.

Regular structures.

Regular structures in patches can be tedious to build and generators such as neural nets (CMJ) and decision trees (Armstrong) are investigated for arbitration. Currently, finite state machines (FSM) are provided to help the operator in monitoring states and triggering the proper events, becoming an extension of his/her gesture. They also can be seen as a compact representation of a conditional network of objects.

3. Performance examples.

Four different performances ranging from installation pieces to dance are briefly described. Each one of those were testing specific lighting ideas and computer control. Some of those have been implemented directly in programming language while others were using the @FL environment.

"Rite of Spring". (rules)

The "Rite of Spring" was performed and adapted by Michael Century for a Yamaha Disklavier and a MIDIGran for the USITT89 in Calgary (Bant). The contrast of mechanical and human performer led to a set and lighting design where pianos were on opposite ends of the studio. The lighting was slowly evolving from one end to the other depending on the musical density of each performer.

Along with this wash, four white stands were set up in the middle of the created diagonal. These were used to bounce amber lights with slow moving shadows around them. While the wash was moving towards a piano and interfering with the shadowing, a balancing behavior was applied around this area.

This was a first experimental phase for a basic set of behaviors. A local behavior around the stands and a global behavior for the balance. Behaviors were simulated before hand and then a large set of cues were sampled from the results. Each cue was then synchronized to the sequence.

"NoJazz noClass". (Cypher's listener)

The lighting software had the chance to be interaced with Robert Rowe's Cypher listener [Row]. The listener outputs a series of parameters from live MIDI input such as: speed, density, register, chords, etc.

For this live music performance, state changes and listener analysis were mapped into groups of lights. It was an occasion to verify the readability of musical gesture via a straightforward lighting scheme. Operation
on the lighting board gave overall washes and some color changes.

"laure gauche". (FSM support)

"laure gauche" was a gathering of video, electro-acoustic music, lights and dance. Each of the medias were controlled live with MIDI custom controllers and keyboards.

The FSM is of great interest in this context. The need to quickly swap from a set of lights to another without having to remember a whole keyboard mapping was critical. The FSM was utilizing that only one note was fired (killing the previous one and crossfading smoothly).

The keyboard mapping had stage up and own references, groups of high side in center stage and some predefined lighting sequences triggering. Sections of the performance were sequenced and played back as background activities while we were operating.

"Létho". (Trash/Shared control)

This installation is an archetypal of memory through an environment of light and music. The piece used 16 RGB controllable lights and four moving light beams. The music was all sequenced and a series of tracks were added to control and synchronize the software on the MacII. The creative process was to deal with a set of high-level control building blocks and assemble the visuals as the sequenced music was playing.

When parts of a device have autonomous control while the human operator gives higher-level control/decisions, interrupts and resists tasks, gives illegal input and generates exceptions, it becomes difficult to structure the system to handle all these circumstances. Data management primitives were of great help to navigate throughout the performance. The building blocks were designed to provide feedback and fail-safe states to reconfigure the system.

For instance, with the ColorPros, the system helps me to build a database of interesting shades. A set of eight were laid out into a circle and the computer was distributing a selected shade from a starting point in the circle. The user or the sequence were giving the starting point and a patch was distributing the shades accordingly. A toggle was used to make the incoming sequence stream. A set of red, green and blue shades were built into data bases during rehearsals and the control patch was utilizing the desired grades and calculating proper offsets to lay the shade in real-time.

There was also a 3x3x3 cube where each axis intersections had an object to bounce off right. Lights were focused all around the cube and patches to highlight these points were created. We had a point patch (x,y,z,interet angle) and a plane patch (vertical, horizontal, side and edge). Slow movement from point to point or plane to plane were built by feeding in parameters to a set of these patches.

With the moving beams, I created a set of "drivers" to get away from numbers. Each beam had an icon on the screen and I could feed them the desired control values. Furthermore, with projective geometry, control position the beam with x,y,z stage coordinates were created. Then I layered MIDI control and I was sequencing the movements of the beams.

4. Discussions and further research.

It is clear that computing power will soon run out. Co-processors are being considered and could lead to an architecture where the MacII runs high-level messages while parallel tasks such as Cypher’s bistar and vector (cues) engine are distributed. Animation tools to speed-up complex crossfades exploration are also investigated. Simplifying synchronization mechanisms and descriptions among messages is another. Robecks and proper overall state of the system is still very cumbersome and needs further structuring.

Involved computer programming is not a task that a designer wants to attack. A fair amount of "what’s going on" is unfortunately needed. Symbolology, languages, production and rehearsals techniques are to be derived, an impact of new devices and tools.