Natural Language Control

as used in

This essay explains the core technology used in Cognito's internal 'fade engine' that makes DMX512 and runs the lights and the end devices at the bits and bytes level. Reading this and understanding Natural Language Control is not necessary to operate the console, but it will give you an appreciation of how lighting control has advanced over the years. Using today's advanced lighting systems has never been easier because of Natural Language Control.[1]

[1] Much of this document was originally published in 2005 by Horizon Control in a white paper called The Abstract Control Model. Horizon was purchased by Acuity Brands Lighting in 2011 and the entire team joined Pathway Connectivity when it too was acquired by Acuity.

Background

Communication and the expression of ideas is central to the art of lighting. Creating great lighting is a team effort lead by the designer. The language a designer uses to communicate with the team, and specifically the console programmer, is crucial to the process of creating the art. The programmer, in turn, must then train the console in order to orchestrate the lights to ultimately relay the intent of the designer to the audience. There is ample opportunity in this process for misinterpretations to muddy the waters of communication. More recently, and at a furious pace, LEDs and multiple attribute "intelligent" lights have entered the mainstream market and the multitude of options they provide has complicated this process amplifying the opportunity for 'miscue' of intent. The simple act of positioning a fader somewhere on a 0 to 10 scale will no longer suffice.

Not surprisingly, there has been an increasing necessity to simplify the process of lighting control. Unlike the hard and fast rules that have existed for decades, a uniform language for designers and programmers to use when describing light behaviors has been non-existent. Moreover, the method used by the console to communicate to lights has never been standardized. The pioneering manufacturers of automated lighting equipment each implemented different philosophies of control. Historically it has been a challenge for some controllers to turn such lights on, get them in a color and make them move about. In all respects, these consoles were merely outputting numbers, sometimes masqueraded by words to get the job done. But now that intelligent lighting is no longer in its infancy a control system that meets the needs of 21st-century lighting fixtures is a welcome addition to the designers' arsenal. Cognito embraces that challenge and makes programming today's complex lighting systems simple again.

Let's go back to the advent of computer-controlled lighting to examine the issues that plagued communication in the theatre. Before computers entered the theatre, the most popular dimmer controllers were known as piano-boards. These large devices had individual handles for each dimmer and designers would ask operators to move a handle to a position to set the light level. These 'move' instructions were written down as cues and with each one executed in succession you had a show. The advantage of this system (which was only realized fully after the obsolescence of piano-boards) was that each move could be controlled at different rates and multiple moves could be executed simultaneously by different operators.

Computer control first appeared on Broadway in 1975 when Tharon Musser used the Electronics Diversified LS-8 console on A Chorus Line.[2] This new technology allowed for unprecedented repeatability and a huge number of cues executed in record time. As processing power was very limited, decisions had to be made on how to execute these fades. The technology and code development tools of the day dictated that each channel would be recorded in each cue. This greatly simplified the process of playing back a show, or more specifically, jumping from scene to scene during rehearsals. Remember, in the old days of piano-boards, getting to any place at random in the show almost always meant starting from the beginning and executing each cue to ensure accuracy. LS-8 and others could do this with ease. Kliegl quickly followed with the Performance and Strand with Multi-Q and Broadway converted to computer control seemingly overnight. Designers were excited by the apparent new flexibility that these computers offered.

These early computer control systems did not emulate piano-boards, but rather manual preset boards. What designers eventually figured out, given a bit of experience on these consoles, was that they could not achieve the complex cue timing that two or three piano-board operators did in the past. As these preset consoles recorded every channel in every cue, they only moved from state to state. This resulted in robotic or non-organic fades. It was only when Strand introduced the Light Palette that the technological problem that plagued these early consoles was finally addressed on a computer (in North America at least).

People everywhere (and since) have praised Light Palette for marrying designers' desires and computer control by using a common language. Almost every controller that has been accepted on Broadway since has used core concepts introduced by Light Palette. With the advent of intelligent lighting, so many more parameters have entered the equation that the language conventions which have evolved are discordant and technologically inadequate. The language must be overhauled. Conventional lighting control just worked in 2-spaces; Intensity and Time. That is not so with moving light control. There are many, many more parameters. Moving light control and solid state lighting have suffered from the lack of a common language for designers and programmers and manufacturers to use. In its infancy, intelligent lighting control stumbled along just managing to keep up with an evolving technology and never experienced the sort of watershed

[2] For a detailed description of the LS-8 and other early computerized lighting control systems see Robert Bell's book *Let There Be Light*
ISBN 9781904031246

event that occurred in the industry with the introduction of Light Palette. The problem was compounded by that fact that industry leaders were extremely protective of their intellectual property. There was no sharing of control protocols between lights and controllers. Each manufacturer vigorously protected the methods they used to control their fixtures and automated systems were sole-source. Only recently has the industry evolved to the point where it has accepted that inter-operability is a good thing and there is broad support for ANSI standards such as DMX512 and the inter-operability it enables.

Talking to the Lights with Bits and Bytes

The earliest forms of computer control, though digital at their core, output an analog signal, typically between 0 and 10 volts. Many architectural luminaires are still controlled this way. The control signal set the lights' output from zero to full intensity. Inside the controller, these numbers were generally stored using 8-bit words, giving 256 steps of resolution. With the advent of moving light systems, the resolution was doubled to 16-bit, providing 65536 steps of resolution. Computers then calculated fades that produced a one-to-one relationship between the 65,000 steps directly to motors that moved the light from, say, pan-stop to pan-stop. This concept persisted for years and, given a specific controller tied to a specific lighting system, pre-programmed shows were reproduced faithfully night after night.

The downfall of this method of control is that these numbers ([0-10], [0-255] or [0-65535]) mean very little in the real world. They are actually only significant when used with very specific equipment. When applied to other equipment, these numbers mean very little at all, and in fact are often meaningless.

Talking to the Lights with Natural Language Control

Natural Language Control's objective is to provide an intuitive programming experience and a versatile control system that when played back can actually provide the operator information about the system it is controlling.

Natural Language Control does this by porting the control to an 'abstract' layer. This has a number of benefits:

1. The 'handles' you use to control LEDs and moving lights are more inline with what you would do to manipulate conventional lighting.

2. The numbers and 'words' you use to build cues will actually mean something. You will have an idea of what you can do with the lights and what is on stage by reading the display.
3. If you have mixed equipment, the methodology you use to communicate is consistent regardless of the protocols defined by the equipment manufacturers. The attribute controls are laid out the same for every and any light.
4. Building a set of looks with one group of lights in your rig can be copied to another groups, regardless of what type of lights they are.
5. The cues you have in your show file can be played back with any equipment allowing you to swap out gear at the last minute if need be.

One of the key things in Point #2 above that bears repeating is that Natural Language Control uses numbers and 'words' to control lighting. One might claim that has been done for years with the use of 'named' palettes. For example, moving lights desks can use labeled position palettes to build cues and the cue displays these 'words' to make it easier to read. Don't lose sight of the fact that palettes, like "Down Stage Center", are just placeholders for a combination of values between 0 and 65535. The words themselves do not mean anything to the desk. They are just displayed on the screen for convenience. In contrast, with Natural Language Control, the words do mean very specific things within the cue structure. Some of the words used include:

- 3200 Kelvin
- 15 degrees of pan
- rotate counter clockwise at 6 RPM
- strobe at 9 hertz
- reset the fixture's driver

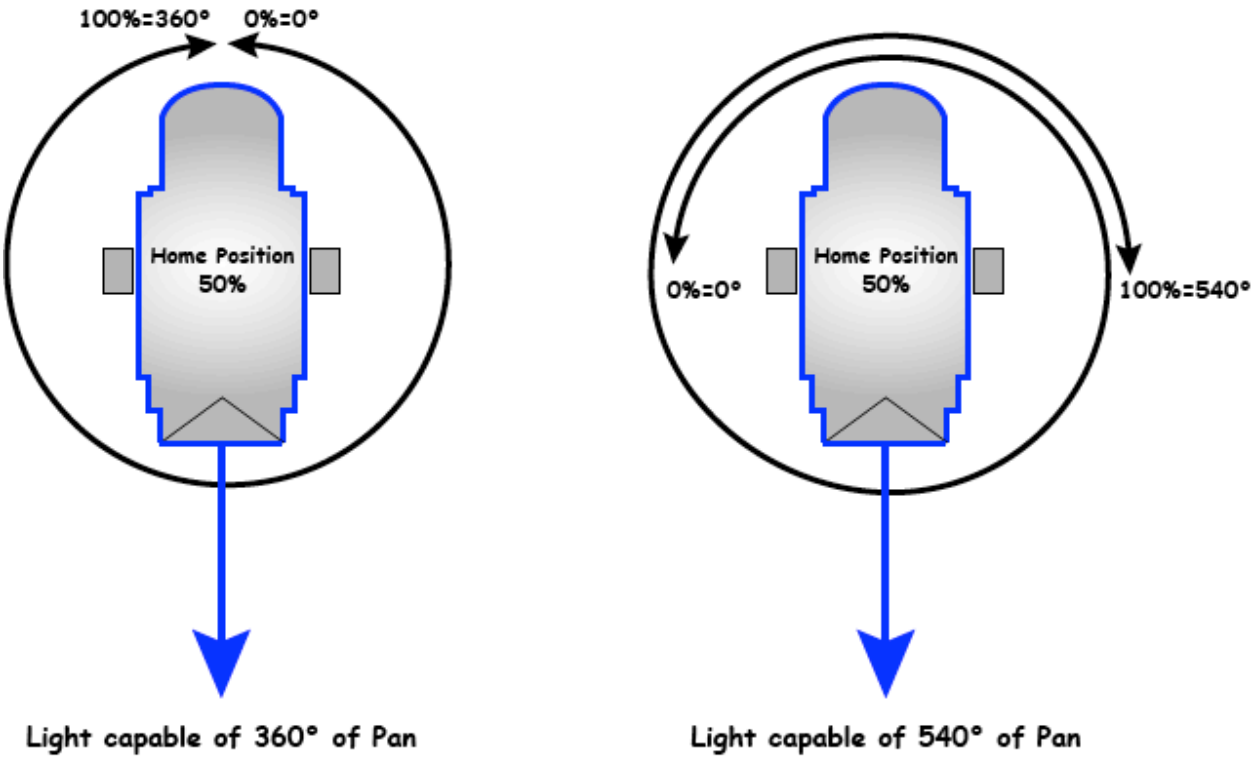
During regular operation, these 'words' need to be converted into 'values' that DMX512 lighting fixtures can use. The trick with Natural Language Control is that this conversion is done each and every time a light is selected, a Memory is recalled or GO is pressed to start a cue (and not before). That means that the protocol, the mode, the model or the manufacturer of the lighting fixture can be changed at any time. Moreover, each and every light, regardless of who makes it, appears similar to the user, giving a more consistent experience when programming the console.

Apart from the benefits described above, this method of controlling lights is not restricted to traditional linear channels mapped to attributes on the light. A few examples below will demonstrate the intuitive nature of describing lights' attributes as opposed to traditional convoluted methods that sometimes group completely unrelated behaviors on the same control channel.

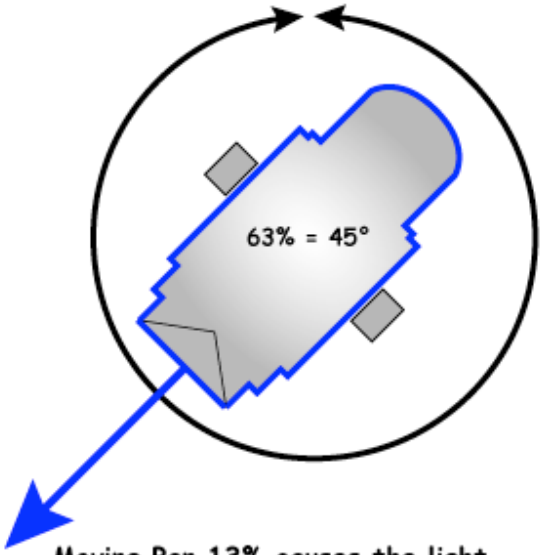
Pan and Tilt Example

The Home position for pan and tilt on most DMX lights is 50:50 (or 32767:32767). This positions the light such that you will have maximum movement in each direction before encountering a pan-stop or tilt-stop. For a light that has a total pan range of 360 degrees, with the control channel set to half, you are sitting at 180 degrees. Taking the control channel to full will move the light 180 off axis towards a stop. So, to summarize, a value of 50% means "go to Home", and a value of 100% means "go to the pan-stop 180 degrees from Home". Figuring out that 90 degrees is half way in between those two values is easy. That would be 75%. And a 45 degree pan from Home is, again, half way between those two values or about 63%. Now it begins to get a little too complex for the programmer to calculate quickly.

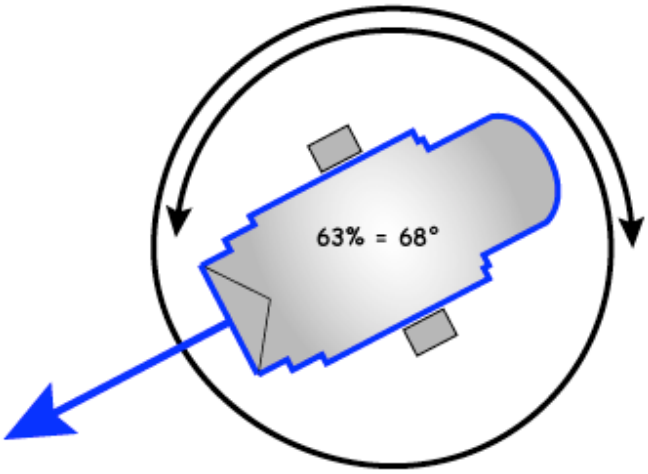
To add to the complication, imagine you have another light in the rig that has a total pan range of 540 degrees.



Now the numbers you just figured out for the first light mean nothing to this one. Worse yet, if you grab them both and pan them in tandem, you would get completely differing results:

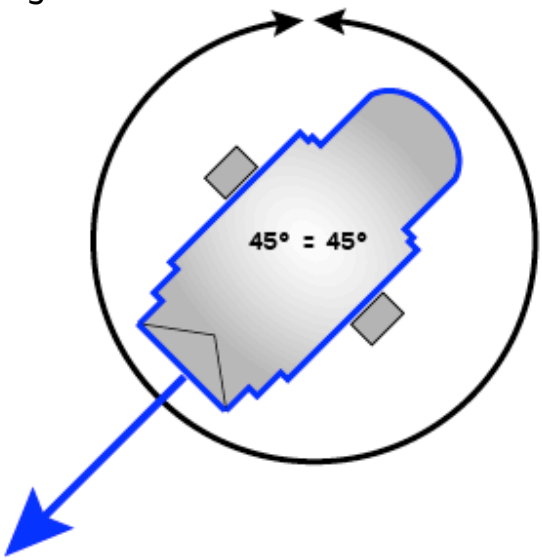


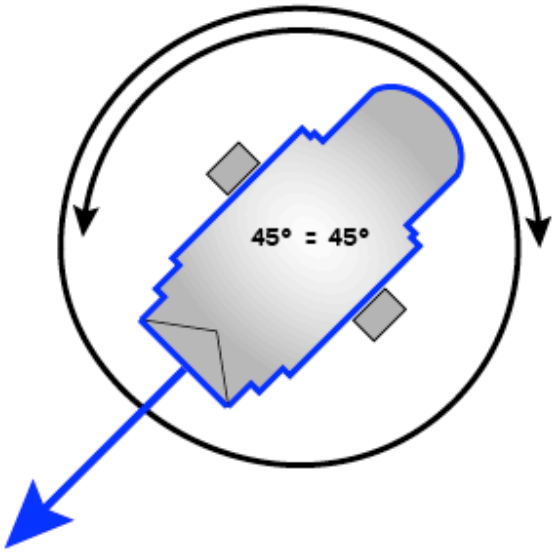
Moving Pan 13% causes the light
to move to 45 degrees from Home.



Moving Pan 13% causes the light
to move to 68 degrees from Home.
The two beams are no longer parallel.

The angles of pan are completely different. The beams of light are not even close to parallel. You can see how this can be very frustrating if you have a mixed rig. With Natural Language Control, the Pan attribute is represented in real-world units of degrees. Therefore, when you talk to the light, you tell it to pan so many degrees:



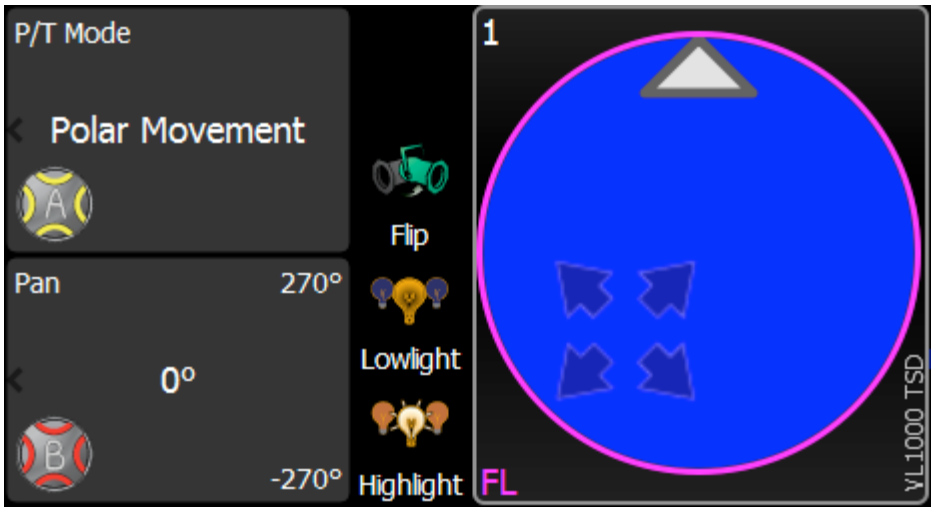


With Natural Language Control, the units used
on the controler are the units used on the stage

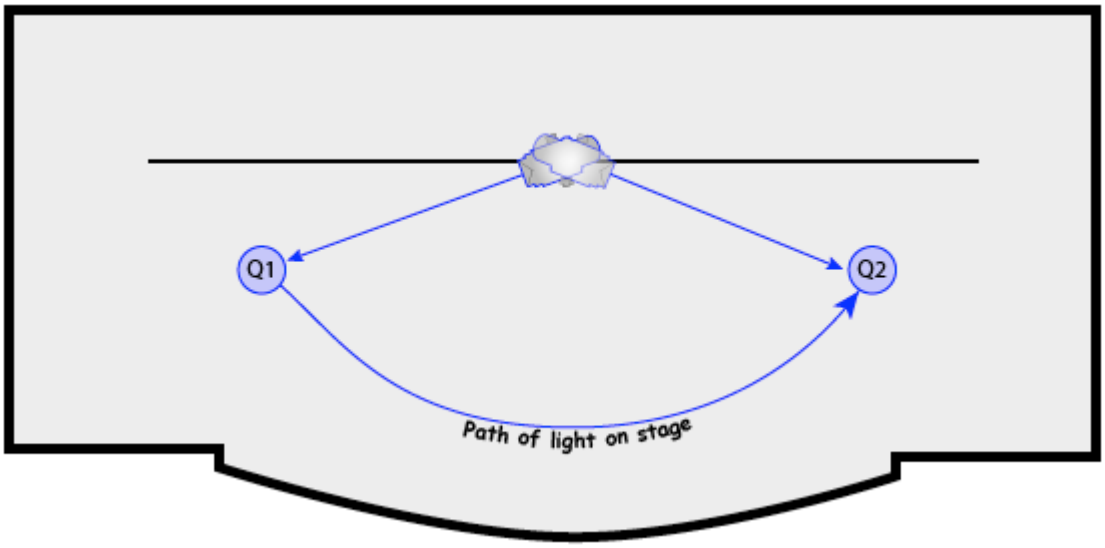
Forty-five degrees is forty-five degrees. This makes controlling a rig that is made up of different types of lights easy to communicate with and easy to understand.

Since Natural Language Control doesn't figure out DMX values until the very last second, it can also alter the way in which the conversion is done at run-time,

producing new and exciting methods of transition during the fade from cue to cue. Various attributes, such as position and color lend themselves very nicely to working in different ways. Color Space is described in detail below, but let's examine how we can move from one place to another on stage given two stored end places.

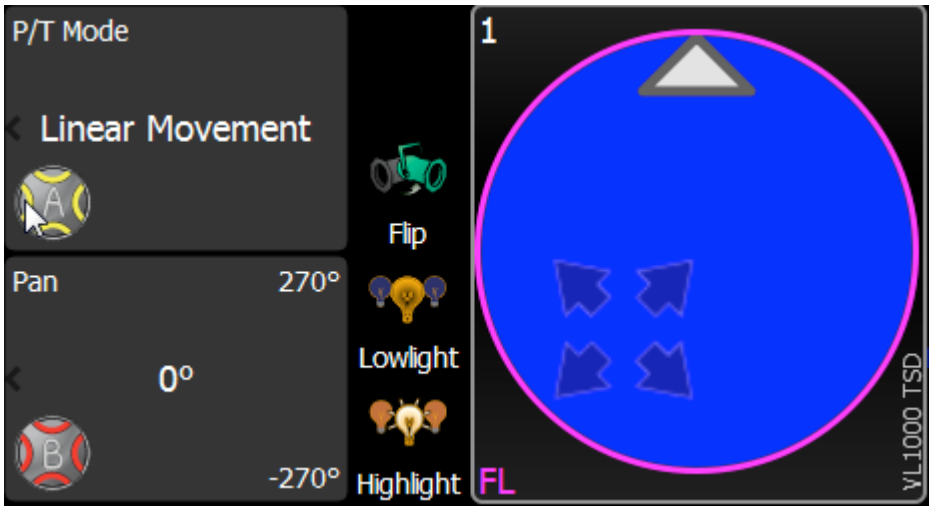


Moving lights achieve movement by physically moving the source with two motors housed within a yoke. This Pan/Tilt relationship equates to a polar coordinate system using azimuth and elevation. When you pan **more** than you tilt the light will move in an arc:

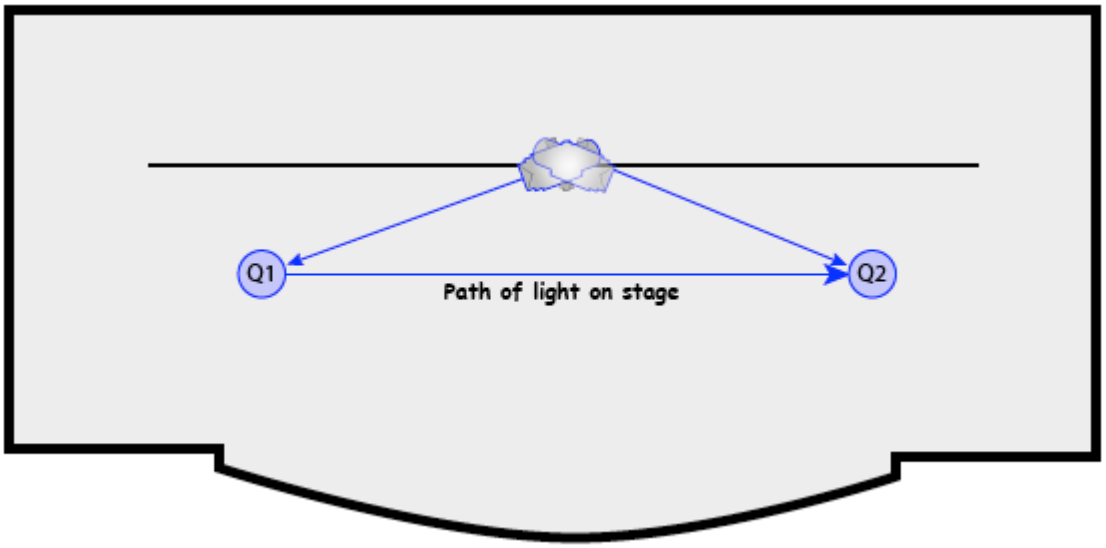


We have become used to this characteristic movement of intelligent lights. Very good moving lights that move extremely smoothly are sometimes described as

moving in an organic manner or looking like they are operated by a follow-spot operator. People are quick to forgive the fact that they are always moving in this arc pattern. Natural Language Control gives you the option of how the light will move. It doesn't have to move in an arc. When a follow-spot operator moves a light from point A to point B, the light normally travels in a straight line.



Cognito has a Positon attribute called P/T Mode that alters the way fades are calculated when you advance from one position to another. If you record a memory or a cue using specific Pan and Tilt values and specify the P/T Mode to be Linear Movement, the end points of the move do not change, but the intermediate steps of Pan and Tilt needed to get from the first position to the second position do change. It becomes a transition that forces the Pan/Tilt mechanism to travel the beam of light in a straight line:

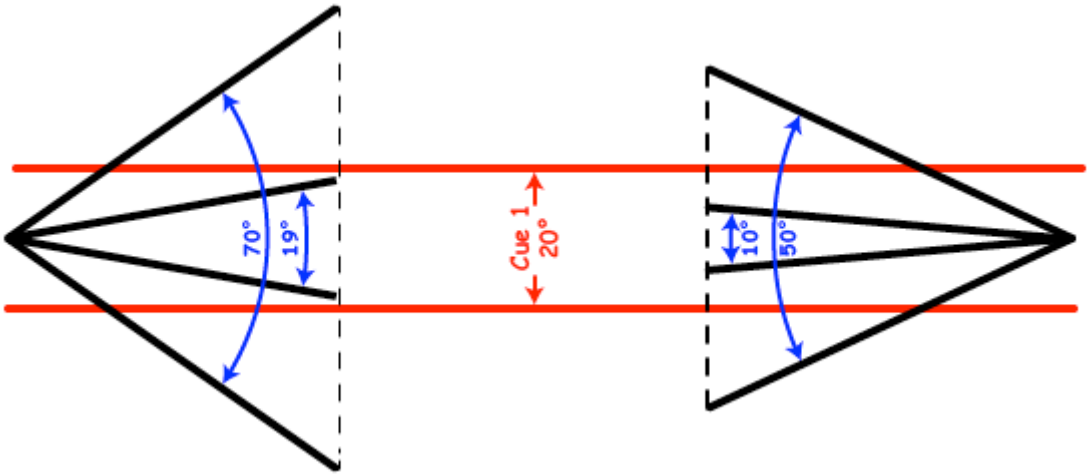


Zoom Example

Programming lights using real-world values allows you to swap one fixture for another and get predictable results. Far more useful is the fact that the same values are used to control different types of lights in a similar fashion. Looking at the zoom attribute demonstrates this again.

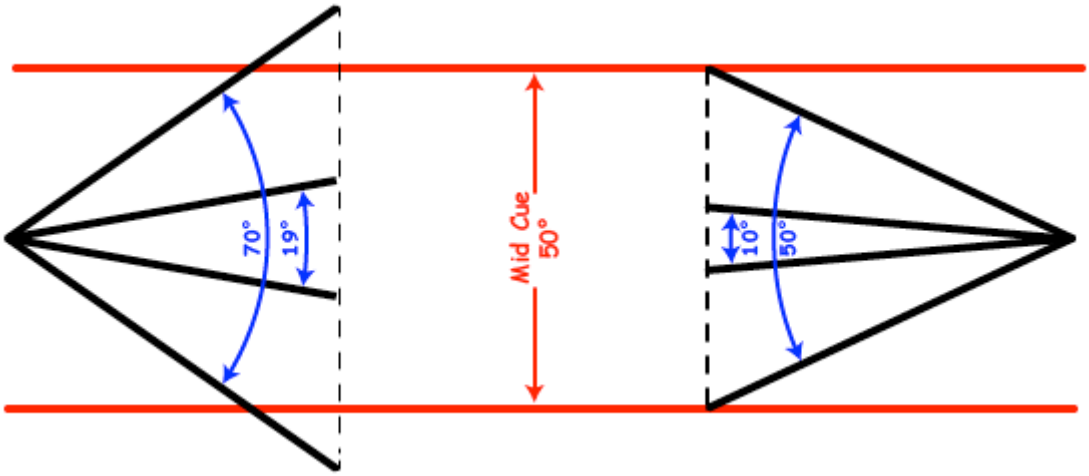
It is quite common to have two or more different types of lights in today's lighting rigs. Matching beam sizes is a process of grabbing one type of light, setting its zoom, then selecting the other and tweaking it to match. You cannot grab both and crank the wheel and hope to get matching results. Natural Language Control eliminates this unnecessary practice.

Here are two lights; one that has a zoom range of 19° to 70°, the other from 10° to 50°. Cue 1 calls for the lights to use a zoom of 20 degrees:

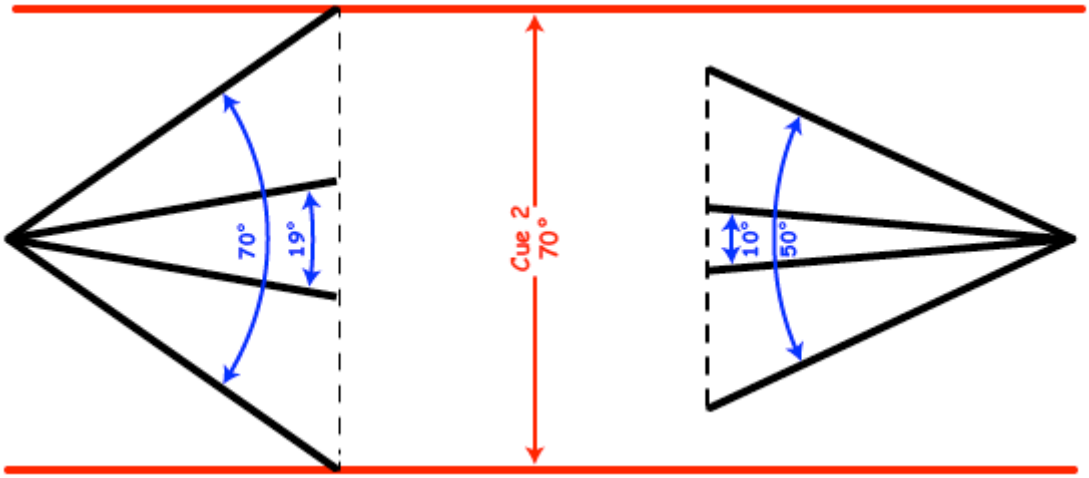


All you have to do is turn them on and Cognito defaults them to the same value of 20°; they're already the same size! Your rig looks consistent and symmetrical with no undesirable surprises and no need for manual re-translation. If you want them to match your 19° or 26° or 36° fixed lights, just set the Zoom value to the appropriate level.

If Cue 2 was written such that both lights go to 70° both lights would resize at the same rate until the one on the right has to give up mid cue:



The light on the left would complete the cue zooming all the way to 70°:

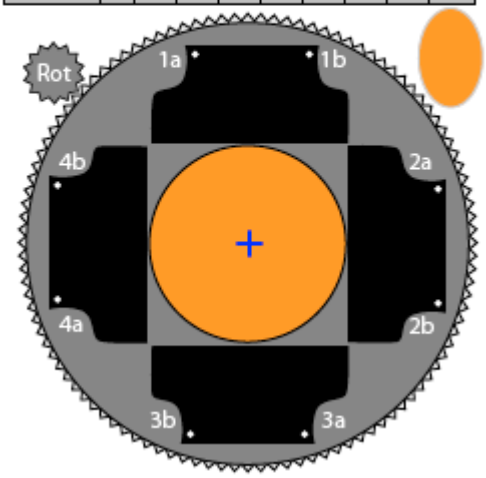


To be fair, Cue 2 could not have been written using the light on the right. This cue must have been recorded using a light that can achieve 70°. Even though in this example it was played back using a 50° light, it does not change the cue. If you later swapped it back to a 70° light, it would go to 70°. It is only when writing cues that you are limited to the physical constraints of the light currently patched.

Shutter Control Example

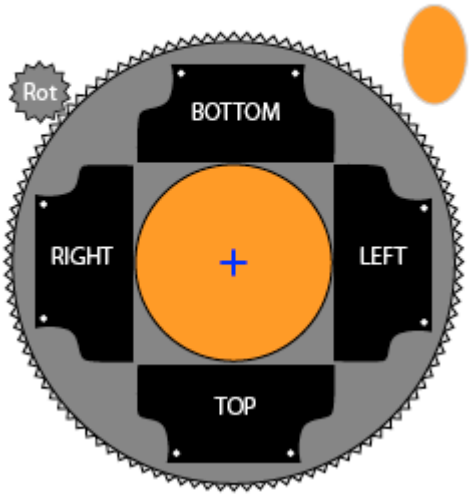
One of the most time consuming endeavors when programming moving lights is shutter control. To achieve desirable effects, the shutter mechanisms need a lot of motors, and hence, a lot of control channels. Typically, most shutter assemblies have nine motors. There are four shutters, each using two motors to control its position within the aperture of the fixture and a ninth to rotate the entire assembly clockwise or counterclockwise. Many times these channels are labeled like this:

Channel	1	2	3	4	5	6	7	8	9
Motor	1a	1b	2a	2b	3a	3b	4a	4b	Rot
Units	%	%	%	%	%	%	%	%	%
Default	0	0	0	0	0	0	0	0	50



The oval in the image above shows how the light would fall on stage if the fixture was hung in a typical Front of House position. You can imagine that trying to make shutter cuts can be a time consuming effort of hunting and pecking for the right channel or more likely, pair of channels. That is why Natural Language Control groups related pairs together into useful names like Top, Bottom, Left and Right.

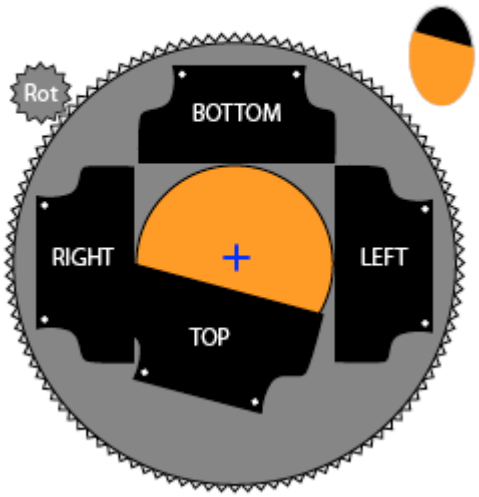
Attribute	Top		Bottom		Left		Right		Rot
Part	Thrust	Angle	Thrust	Angle	Thrust	Angle	Thrust	Angle	
Default	0%	0°	0%	0°	0%	0°	0%	0°	0°



In the example shown below, the console is driving a Vari*Lite 1000 fixture. Cognito conveniently places the thrust controls on the two wheels closest to you

(red and green) and the respective angle controls above those. One wheel manipulates motors 1a and 1b in unison to Thrust the shutter into the aperture of the fixture. The yellow wheel above that adjusts the relationship between those two motors, giving you one handle for controlling the Angle of that shutter. If you rotate the yellow wheel to the right, the shutter rotates right:

Attribute	Top		Bottom		Left		Right		Rot
Part	Thrust	Angle	Thrust	Angle	Thrust	Angle	Thrust	Angle	
Value	25%	15°	0%	0°	0%	0°	0%	0°	0°

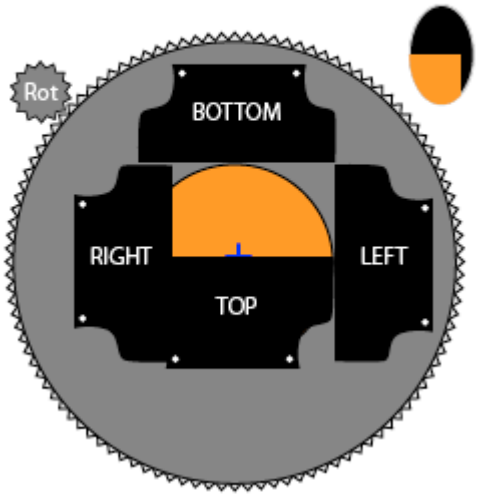


Note that the Thrust is measured in percentage. Most moving lights only allow you to put the shutter part way into the field of light. Above you can see the Top Thrust for the VL1000TSD is set to 25%, but its maximum value is 80% (not 100%). That is because the VL1000 physically can only cut out 80% of the beam. The angle is limited to -35° to 35°. Conversely, the VL3500 can only push the

shutters 47% of the way into the beam. By using a percentage unit to describe the position in the aperture and degree units to describe the angle, you can copy the shape of a shutter cut from one type of moving light with one set of physical constraints to another with predictable results.

In the example below, the value of Top Thrust is set to 50% and the Top shutter is cutting the beam in half. The Right Thrust cuts 15% of the beam.

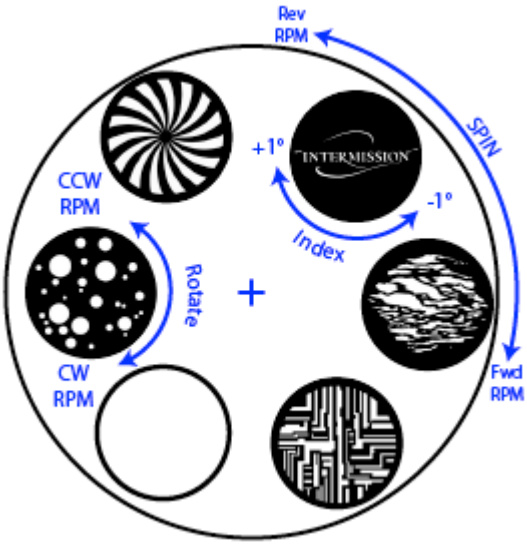
Attribute	Top		Bottom		Left		Right		Rot
Part	Thrust	Angle	Thrust	Angle	Thrust	Angle	Thrust	Angle	
Value	50%	0°	0%	0°	0%	0°	15%	0°	0°



Gobo Control

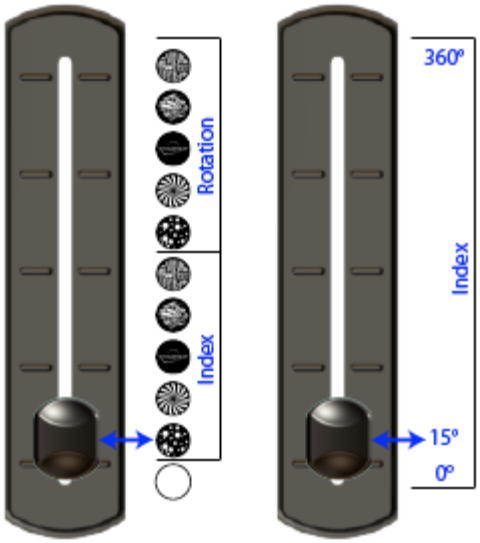
Individual moving light manufacturers' implementation of gobo control is frustratingly inconsistent. There are so many things these modern machines allow us to do, but there has never been a consistent method of describing what they do. Natural Language Control attempts to pull in the reins and consolidate on a common language of control.

The assembly that holds the entire gobo selection is called the **Wheel**. Wheels can **Spin Forward** or **Reverse** in Revolutions per Minute (**RPMs**) or can **Select** individual **Gobos**. Gobos can be **Indexed** in **Degrees** like hands on a compass or **Rotated** continuously **Clockwise** or **Counterclockwise** again at a specific **RPM**.

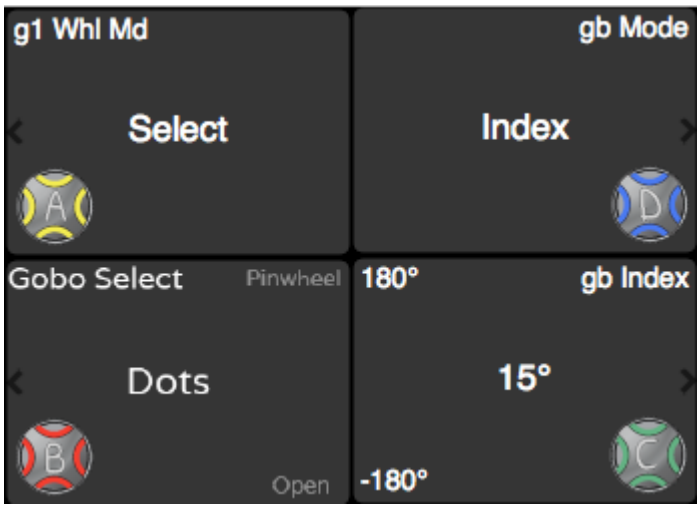


Different manufacturers use a variety of control channels to achieve all of these possible behaviors. Some use lots of channels which surprisingly makes the control of the gobo wheel easier, and other insist on bunching up behaviors on only a couple of channels. The examples below are generic and are only used to show how it could be done using linear DMX512 channels compared with how it's handled using Natural Language Control.

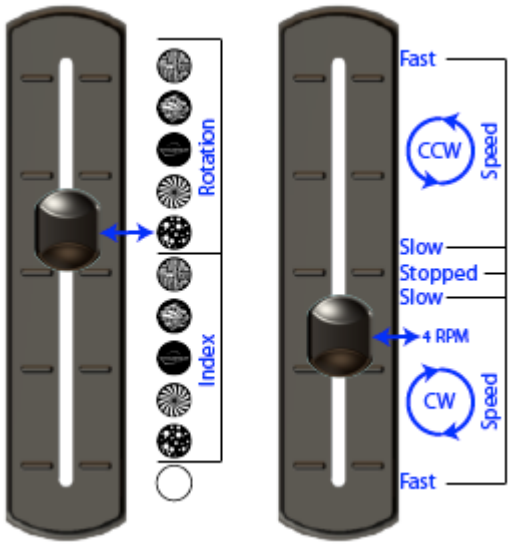
The first of the pair of these linear channels is used to position the wheel and select a specific gobo and do one of two things with it; either Index it or Rotate it. The second channel changes modes based on the position of the first. Here the first channel is set to about 10% and Selects the Dots gobo for Indexing. The second channel is set to about 10% which indexes the gobo 15°.



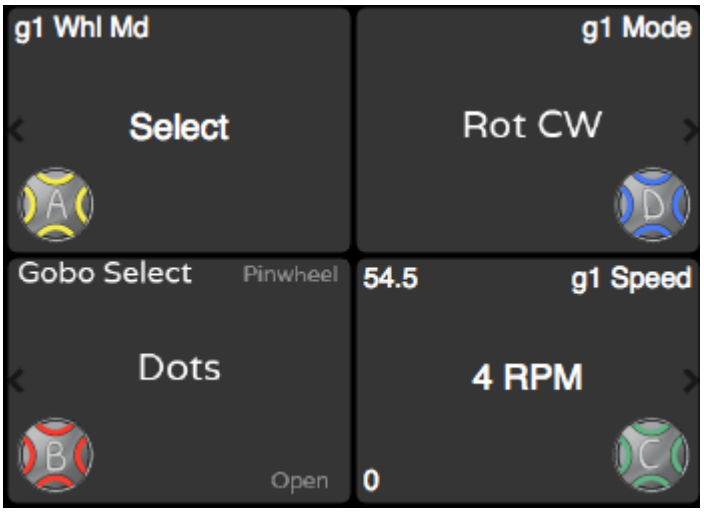
Remember, the fader doesn't show you these options as selections - **you need to know them in advance!** In contrast, Cognito's display shows Gobo 1 Wheel Mode (yellow wheel), Gobo Select (red wheel), Gobo Mode (blue wheel) and Gobo Index (green wheel). And, as well as showing "Dots" as the current selection, you can see that moving one 'tick' forward would give you "Pinwheel" and one tick backwards would give you "Open":



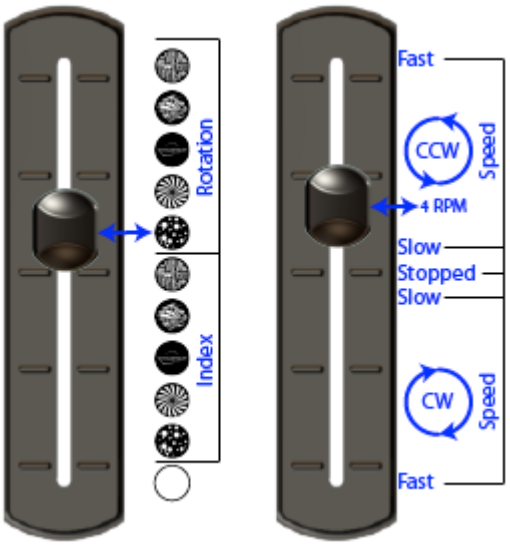
To rotate the Dots gobo continuously in the clockwise direction, the first handle must be placed at 60%. That changes the mode of the second handle and the 10% position is now meaningless. To see a rotation of 4 RPM clockwise, the channel must be set to 30%. (Where that value comes from is completely arbitrary. Truthfully, you would never get a answer even if you were to visit the factory and ask the firmware engineers!)



The same information is shown on Cognito's display like this:

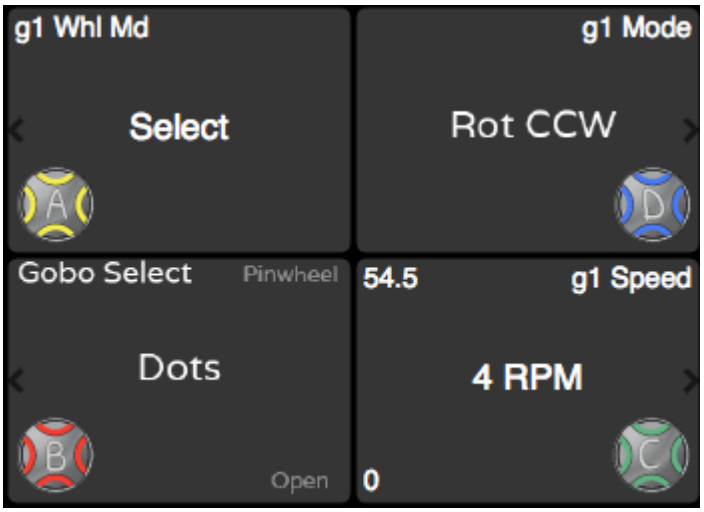


Changing the direction of the rotation on a DMX based system means you must travel the second handle through a bunch of values that are of no interest to you. The gobo would slow down, then stop, then change direction and speed up again as you adjust the control channel. This can be very disconcerting for a designer who is watching the stage. None of those behaviors were asked for, but were necessary to reach the desired result.



With Cognito , you just nudge the blue wheel one tick to change the value from **Rot CW** to **Rot CCW**. The green wheel, which controls the speed, is not changed. The DMX values will jump from the value for clockwise rotation at 4 RPM directly to the counterclockwise 4 RPM value (whatever that is).

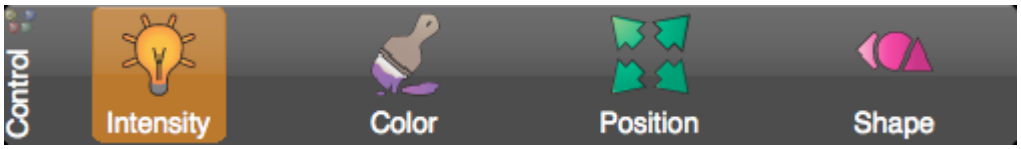
Again, this is how it looks on Cognito's display when programming:



Conditional Abstract Attributes

Automated lights are riddled with control parameters. In earlier days, many fixture manufacturers combined DMX512 channels to achieve separate effects in an attempt to prevent the fixture from consuming an outrageous number of channels. A common practice is to use one channel as a mode channel to modify the behavior of another. This makes life difficult for the lighting programmer as he never knows what a handle will do without first checking the state of the mode channels.

Natural Language Control eliminates this need for reference by not presenting you with controls that are ineffective on one channel because of the state of another. The Control Task separates the tools into one of four attribute families:

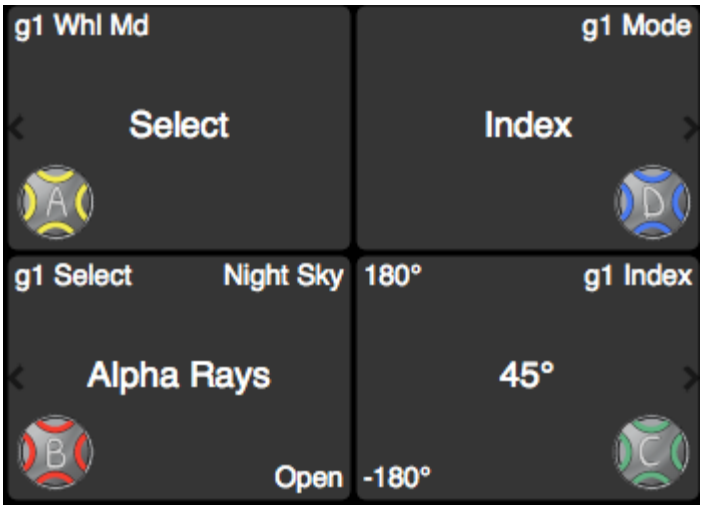


- **Intensity** (amber colored tools)
- **Color** (purple colored tools)
- **Position** (green colored tools)
- **Shape** (pink colored tools)

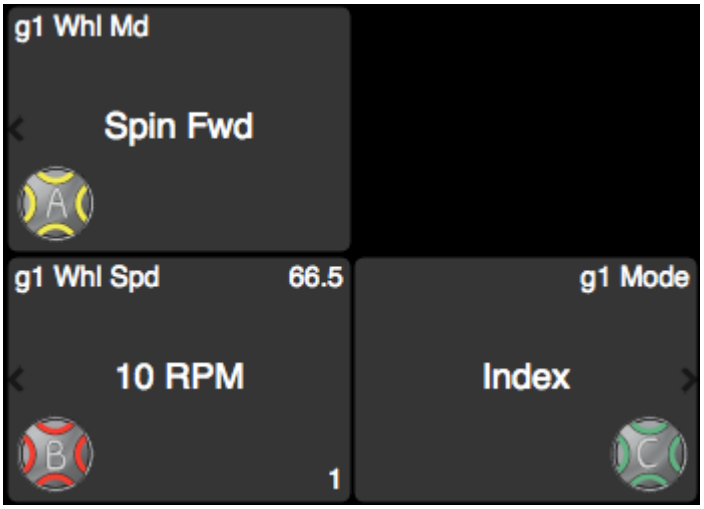
The various tools in the Control Task allow you to adjust attributes without the fear of inadvertently affecting the function of another channel. Furthermore, the

Wheels tool, which is common to all attribute families, labels each control appropriately as to what it is doing at present based on the state of other attributes. Complex gobo wheels are a good example of how this is put to use.

DMX512 mapping and the number of DMX512 slots used by the light have nothing to do with how the controls are laid out to the user.
In this example, Gobo 1 Wheel Mode (yellow wheel) is dialed to **Select**, the current Gobo Selection (red wheel) is **Alpha Rays,** the Gobo1 Mode (blue wheel) is set to **Index** and the gobo has been Indexed (green wheel) to 45°:

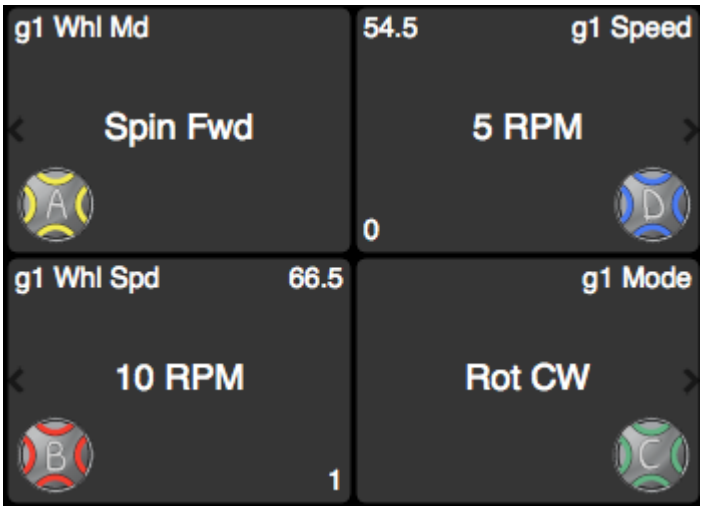


If the user were to adjust the control labeled Gobo 1 Wheel Mode (yellow wheel), the entire gobo wheel would **Spin Forward** (or Reverse). If so, displaying the control that allows you to choose the Gobo Selection would be pointless; the entire wheel is spinning through the light path, so you will see all the gobos, not one at a time. The layout is changed to reflect this:

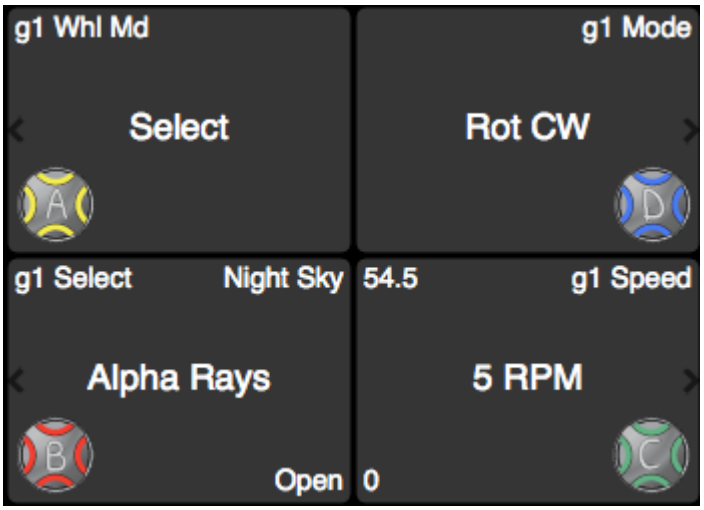


Shown on the red wheel is Gobo 1 Wheel Speed which has been set to **10 RPM**, and green wheel has Gobo 1 Mode set to **Index** preventing the individual gobos rotating. This is where Natural Language Control keeps you out of trouble by only showing you what's possible. The planetary rotation of the larger wheel makes the individual gobos on the smaller cogs appear to rotate around the larger wheel's center axis. Mechanically, you can't prevent this from happening. That is why the Wheel Mode is **Spin** and the Gobo Mode is **Index** you are not shown the corresponding Index attribute (which was set to 45° above).

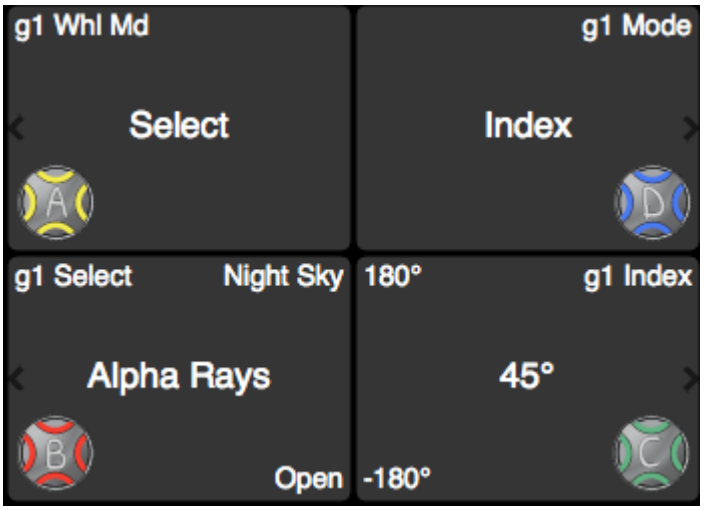
When you do want a lot of gobo action (gears turning in gears), set the Gobo 1 Mode attribute (green wheel) to **Rot CW** (or CCW), and appearing on the blue wheel you will see a new Gobo 1 Speed attribute (blue wheel) which is set here to **5 RPM:**



Changing Gobo 1 Wheel Mode (yellow wheel) back to **Select** will once again allow you to choose a gobo with the lower left control and in fact Cognito has 'remembered' that the last gobo you selected was **Alpha Rays:**



Cognito does keep the fact that you were rotating the gobos at 5 RPM, but if you set the Mode (blue wheel) to **Index** and Cognito will remember that Index was last set to 45° and you're back where you began:

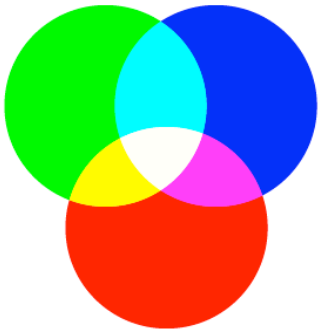


Attribute Substitutions

Copying and swapping attributes among lights that share scalar properties like Position and Zoom is only the tip of the iceberg when using Natural Language Control. The real power of Natural Language Control can be seen when you start using similar, but not identical attributes and how Natural Language Control works with them. Color is a great example. There are three primary automated color systems in use today; Subtractive Color Systems like CMY, Additive Color Systems like RGB (or RGBA etc.) and Fixed Color Systems that use gel (like scrollers) or dichroic glass (like color wheels). Natural Language Control works

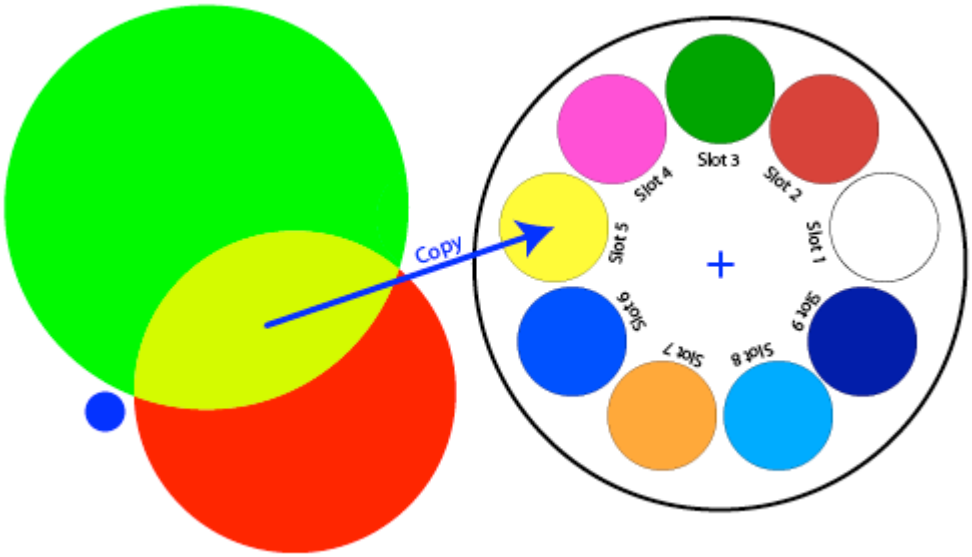
with any combination of these three and make intelligent substitutions between them if required.

The proliferation of LED lighting has made common the RGB color space of additive color mixing. With the three primary colors you can mix them to make white or each at various levels to make colors.



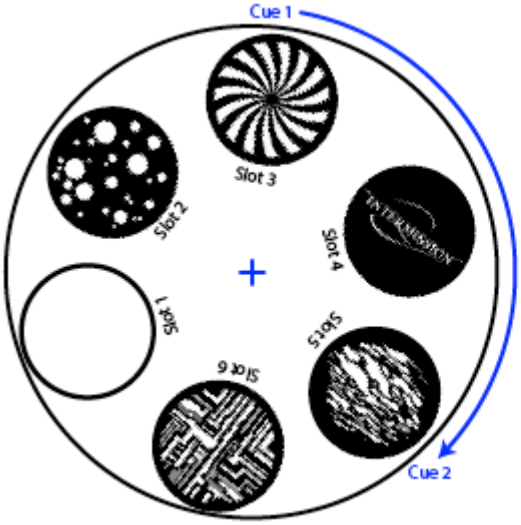
More recently manufactures are using more than red, green and blue to achieve 'better' whites by adding either white or amber LEDs. We'll discuss what to do with these later on. In contrast, traditional stage lighting starts with white light and, by various mechanisms, color filters are placed in the path of the light to subtract out different wavelengths. By introducing varying amounts of filtration, a variety of colors can be produced.

Either way (additive or subtractive), the concept is to mix various colors to achieve the desired results. You can easily work with RGB and CMY simultaneously, referencing either or both in any base color space. If you copy those base color attributes to a light that doesn't have color mixing abilities, but instead has a color wheel, a suitable substitution will be made. Since the Natural Language Control Fixture Library stores a lot more data than just the name of a color, mathematical matching can be done. For example, with the color mix below on the left copied to a light which has a color wheel as shown, Slot 5 would be chosen.

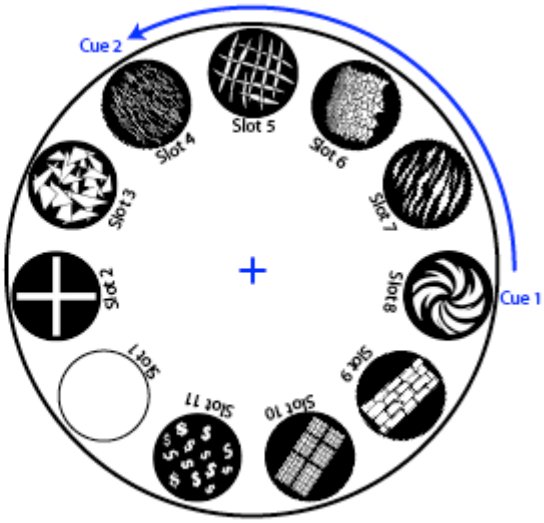


By the way, the color value is not stored as Slot 5, it is stored as "Yellow" so that if it's ever applied to a different light with a different wheel (or mixing system), it would produce the correct or near correct color again.

Gobos are also problematic when using stock fixtures. Different lights use different numbers and types of gobos. If you're touring with Cognito but not with your own lights and you travel from one venue on Thursday to a different venue on Saturday, Natural Language Control can greatly reduce the amount of work needed during Saturday's focus session. Here the cues are written such that Cue 1 uses a Pinwheel gobo and Cue 2 a cloud breakup. On Thursday's rig Cue 1 would use Slot 3 and Cue 2 would fade to Slot 5. As the cues are written and stored based on what comes out of the light, not what protocol (bits and bytes) the light is expecting, we know what Cue 1 is supposed to look like. That is why on Saturday night, Cognito would produce DMX512 values to move this venue's lights to Slot 8 and then to Slot 4.



Light used on Thursday night in Dallas



Light used on Saturday night in Austin

Phantom Abstract Attributes

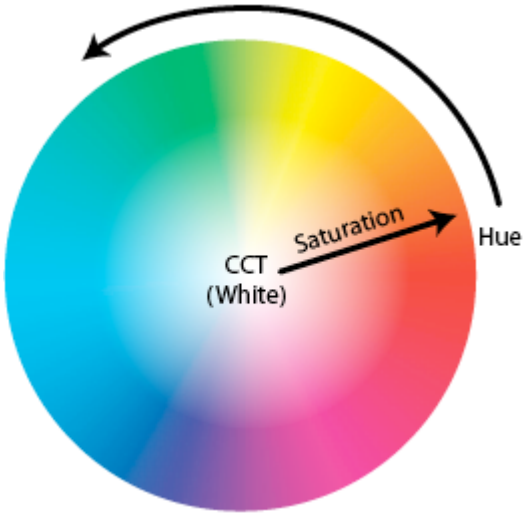
Natural Language Control's ability to provide additional attribute controls the fixture manufacturer didn't allow for makes controlling some types of lights more convenient. An ideal example of this is RGB LED fixtures. Traditionally, the only controls the users can adjust are Red, Green and Blue. Distinctly absent is an Intensity attribute. The undesirable side effect of this is that if you pull down the Grand Master, the LEDs intensities remain unchanged. To blackout the fixtures, you have to adjust three parameters rather than one. This becomes inconvenient when you just want to lower the lights' level or take them out in one cue only and restore them in the next. You must look up the RGB values from the preceding cue to restore them.

Natural Language Control adds a phantom attribute to RGB lights. This intensity attribute does not appear any different from that of any other lights, but it does control the overall brightness of the LED without affecting its Hue. Another benefit of having this attribute is that if you ever do replace an RGB fixture with another, more traditional type of fixture, the Fade Up/ Fade Down dynamics of the light are already in the cues.

Another attribute that is gaining more and more significance, especially in architectural applications is the color of white. As we know, white is made up of all the colors in the rainbow, but if one is more dominant than the others, it can tint the white towards one end of the spectrum or another. Using a Phantom Abstract

Attribute, Natural Language Control can automatically 'tune' your white or Correlated Color Temperature (CCT) when you're not mixing rich colors.

Ignoring brightness for now, there are two parameters that define the color. **Saturation** is the amount of color or perhaps the distance from white, and **Hue** is the dominant wavelength of the light defining its color. Hue is measured in degrees where 0° is red and 180° is Cyan. We measure Saturation in % where 100% is pure color and 0% is white (all the colors).



White is relative though, which is why you 'white balance' a camera or you set the white on your computer monitor. Our eyes can be fooled into believing almost anything with very little saturation (or lack of a dominant wavelength) is white. White light is defined with a combination of colors or a Correlated Color Temperature measured in Kelvin. Kelvin, like Celsius, measures temperature, and in this case, it's the temperature of a white-hot 'black body' that emits light because it's so hot. When it's not so hot, it's closer to the infrared end of the spectrum (red hot poker) and when it's really hot it's blue (like an arc welder or the sun). Household lights are about 2800K, theatrical lights are 3200K and daylight is about 5600K.

Natural Language Control presents a Phantom Abstract Attribute called CCT and its units are Kelvin. If you choose a color capable light, you can dial in the CCT you want the light to make when its Saturation is approaching zero. In this example, the CCT is set to 5600K and Cue 1 is a Blue and Cue 2 takes the light to white:

Equally you could run the cues again with the CCT set to 3200K and it might look like this:

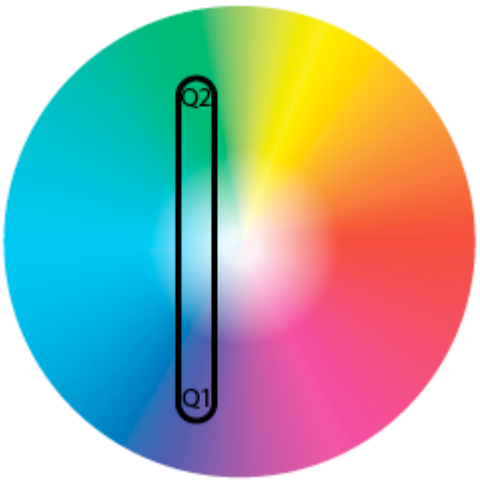
Another option offered by Natural Language Control is the ability to automatically lower the CCT when you dim the lights. This is called Dim-To-Warm. When tungsten lights are dimmed, they cool off (which makes the light 'warmer' or more red). This is appealing and we've become accustom to this behavior. Think of a restaurant at night; they dim the lights to match the intensity and color of the candles on the table. Some advanced solid state lights offer this options internally, but with Natural Language Control you can use this feature on any color capable light making your modern lighting system have the same organic and subtle feel as older dimmer systems.

Color Spaces

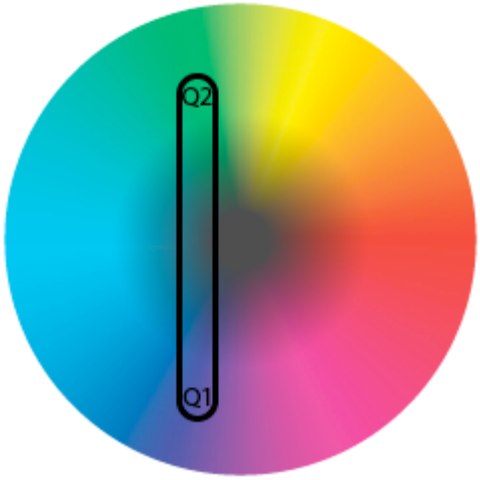
Complimentary Color Spaces are basically different methods used to describe the individual components that make up what the eye perceives as color. None of them are right or wrong. They are individual and each one has its purpose. The selection of one over the other is primarily a matter of choice. Traditionally, consoles only allow you to work in the color space native to the light you are controlling. That is, if you have Red, Green and Blue LED lights, you would be working in the additive RGB space. If you have RGBW lights, you'd need to work with four colors. Working with five or even more sources can get out of hand quickly when we're all used to a tristimulus world. White-sourced lights mix colors with Cyan, Magenta and Yellow dichroics and work in the subtractive CMY space. Some solid state lights actually expect attribute data in Hue, Saturation and Luminance and this can be problematic when you want to fade from a hue of 5% to a hue of 80% as this will show you most of the rainbow; not very subtle! Even though some consoles have color-pickers that allow you to quickly grab any colors, at the end of the day, they generally fading from one triplet of CMY (or RGB or HSL etc.) to another when they run cues.

Natural Langue Control allows you to choose *and fade* in different color spaces: RGB, CMY, HSV (Subtle) and HSV (Rainbow). Having the ability to choose ANY light and work in these spaces is a real benefit because in the old days of DMX512 controllers, you would have to first pick the RGB lights and set them with your wheels, then pick the CMY lights and set them. If you mixed the two types of color systems, they 'fought' each other and doing that can quickly destroy the art on stage. Choosing a color is one thing, but fading from one scene to another in different color spaces can unify a diverse lighting rig with better synchronicity than ever before.

This is how a fade looks in the RGB space fading from purple to green. Mid way between Cue 1 and Cue 2, all of the RGB LEDs come on brighter than when producing a saturated color and the fixture 'bloom' towards white (less saturation).



When you start with white light and introduce CMY flags, midway between Cue 1 and Cue 2, most of the dichroic glass is in the lens tube sucking most of the color (and intensity) out of the light and you 'dip' toward mud.



Regardless of what color system your fixtures use, if you fade in the HSV color space from, say, purple to green where the saturation is pretty much unchanged, the only attribute that moves is Hue.

This sort of fade avoids white and mud and looks more natural on stage. Note that the Natural Language Control decides to fade clockwise around this color space. That is because going through blue seems natural when going from purple to green. We call this a Subtle fade. If we went from pink to amber, it would fade through red. It always takes the shortest path between color or the path that produces the fewest color changes.

If you want a more dynamic effect when switching colors, you might choose the HSV (Rainbow) color space. You still define Hue in degrees (for example, cyan is 180 degrees) but when it fades it takes the 'long way' around.

Conclusion

For years the language and control structure used to control lighting systems has been imposed upon designers by the equipment manufacturers. This was not conducive to an enjoyable experience for anyone involved in the process. Natural Language Control defines a common language that designers and programmers can share and the complex processes of translating this language into DMX or any other control protocol is taken care of for you.

This 'language' has not been defined arbitrarily or in a vacuum. We use colloquial terms that have been used in the theatre for years and present them to the programmer with sense and order. Confusing mode channels that change the purpose of other channels have been eliminated and new and uniquely useful control handles have been added. Natural Language Control allows the designer to look at the lighting rig as a unified tool to aid in the design process. The designer will no longer need to conform to the language of the engineers and this allows them to diversify the lighting rig without the worry of adding complexity to the programming process. Every light now speaks the same language and increased communication and understanding will only lead to better lighting.