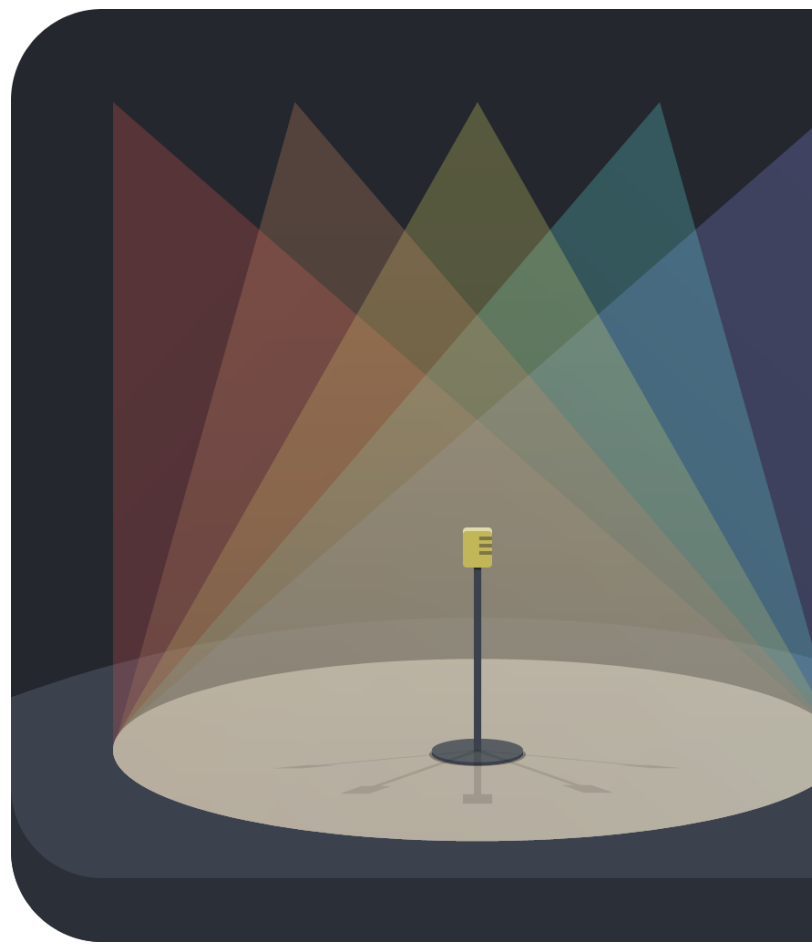


# MaizeDMX 3

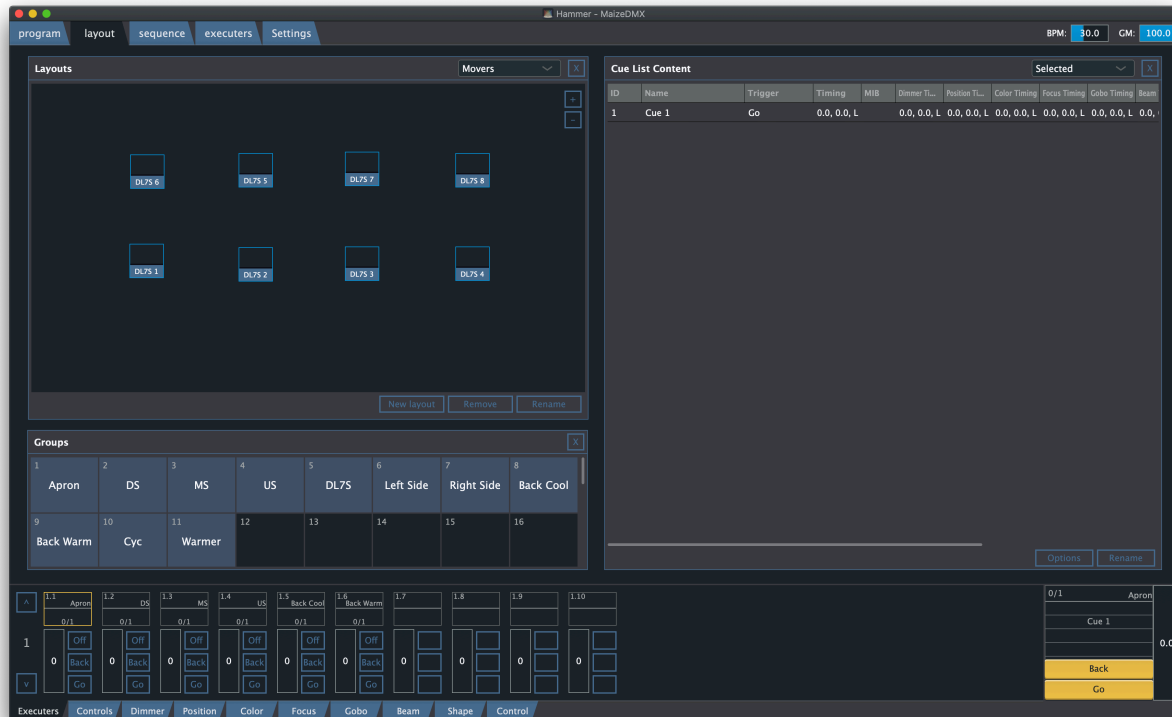
## User Manual



MaizeDMX is a simple and easy to use DMX lighting control software. As beginner lighting operator or designer, you will find it's perfect for your stage or event.

- Supports both ethernet and USB based DMX interface.
- Supports both macOS and Windows.
- Standard concepts like programmer, group, preset, cuelist and effect.
- JSON fixture profile that you can write.
- Sync audio with lighting cues with the sequence.
- Supports MIDI controller with javascript.

# Overview



The MaizeDMX 3 main window is divided into two parts. The top part is called workspaces, the bottom part is for controls.

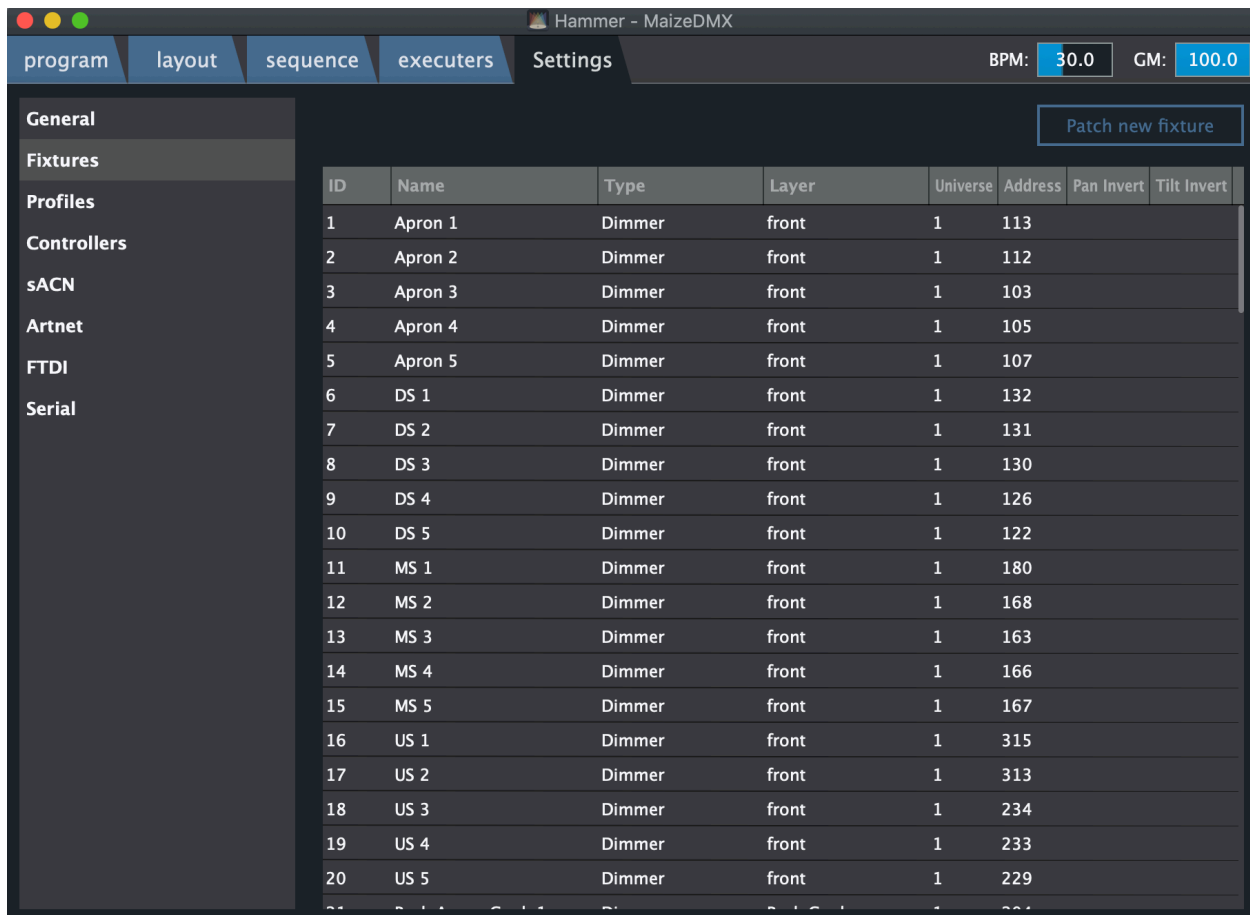
You can create a number of workspace tabs on the top part. In each workspace, right click on the empty area to add feature panels. You can resize and position the panels any where you like. We will talk about different kind of feature panels in the following chapters. (Use F1-F8 to quickly switch workspace tabs)

The control tabs at bottom show executors and parameter controls for selected fixtures. (Use Alt + F1-F8 to quickly switch control tabs)

Now imagine you are creating a new project for a show, let's go through each component of MaizeDMX in a tutorial style.

# Settings

The first tab you want to go is the “Settings” tab, which will always be the last workspace tab.



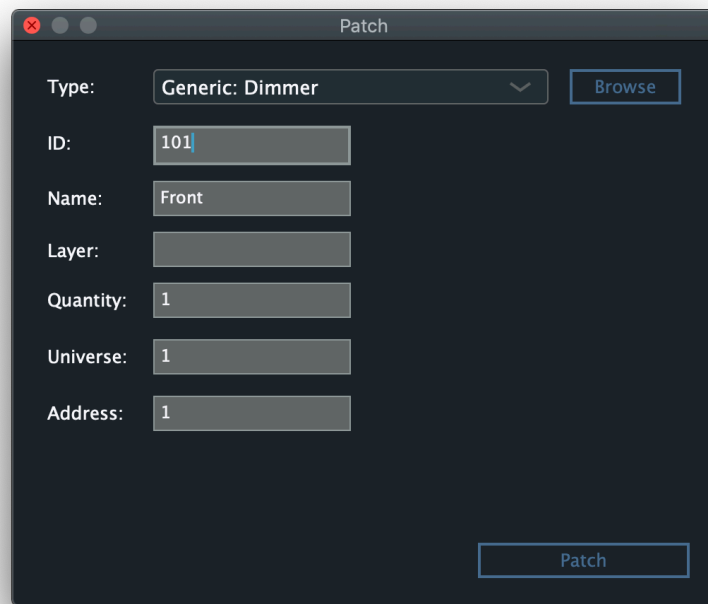
## Output

To really control fixtures with MaizeDMX, we support several different ways to send DMX control data to fixtures:

- USB based DMX adapter: You can find these Enttec compatible devices under [FTDI](#) or [Serial](#). If you don't see your USB DMX device there, please install usb to serial drivers from [here](#).
- Ethernet based DMX node: You can configure those under [Artnet](#) or [sACN](#).

## Patching

To patch (add) fixture to your project, click the “Fixtures” page on the left, and then the “Patch new fixture” button on the right.



In the fixture type combo box field, you will only find dimmer which is the only fixture type that is included with a new project. In order to add other types of fixture, you have to click the browse button and locate its profile which is a JSON file. You can download a collection of profiles from our website, but it won't have everything in the world. If you have a specific fixture to add, you may need to write it. (Don't worry, it's not that hard)

The ID is the number that you will refer this fixture when programming, it has to be unique for each fixtures.

The layer text field is a way to group fixtures together. Just write whatever you want there.

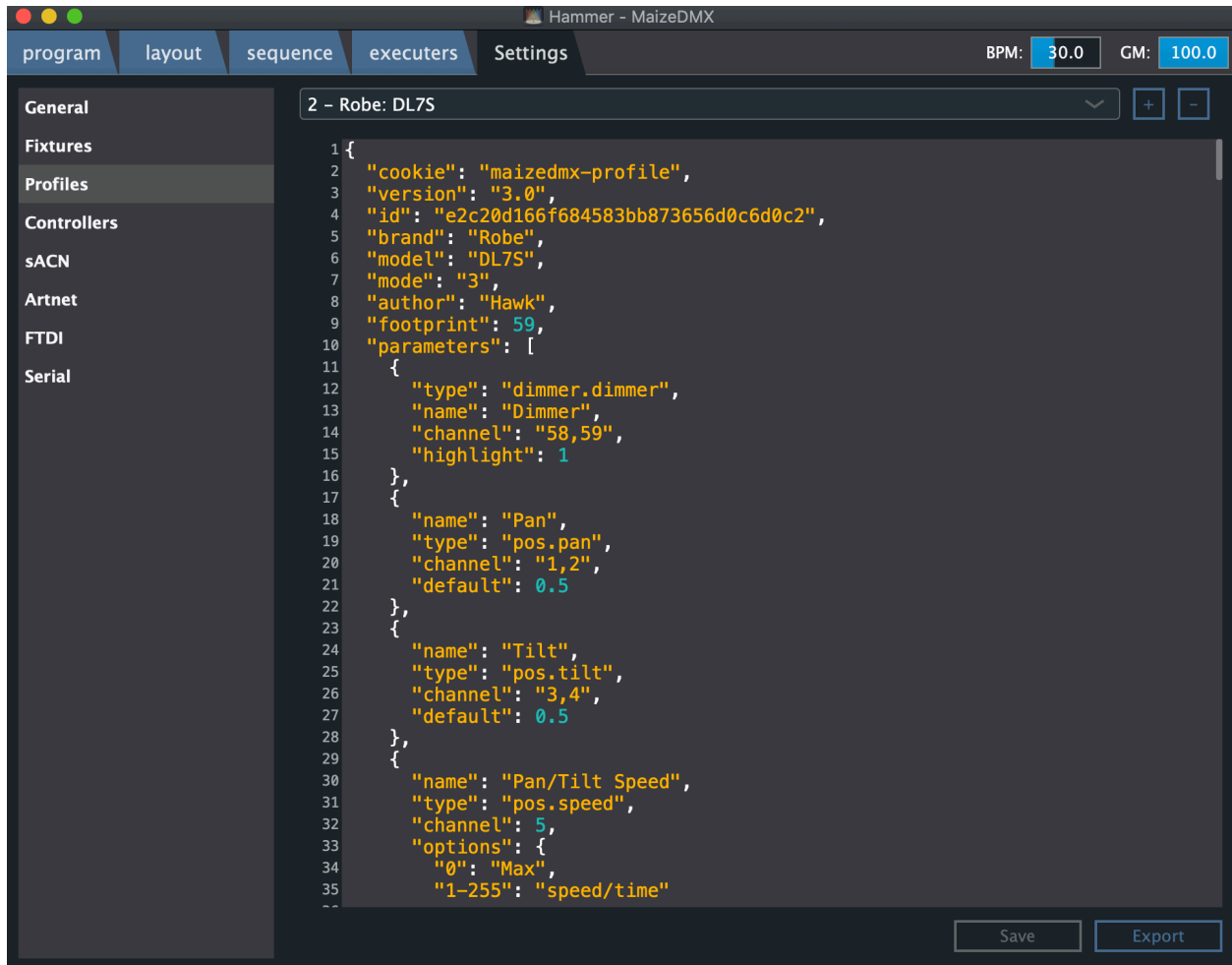
Quantity means how many of this kind of fixture you want to add, MaizeDMX will increase the ID, name and address automatically to differentiate the fixtures in this batch.

Universe and address is the DMX address that this fixture has. Make sure it is the same as given on the fixture.

After clicking patch button, you will see your fixtures in the fixture table. You can right click on a field to edit its value anytime. Remember that you can also select multiple rows and edit them at once.

## Profiles

Want to create your fixture DMX profile now? Click the “Profiles” page on the left, then you will see a JSON editor.



You can write your profile in our JSON format here or even better use an external text editor to write it there so it can be saved and reuse in another project. See appendix for a list of parameter type.

## Fixture List

After adding your fixtures in settings tab, switch back the first workspace. You should see a fixture list panel. It shows your fixture and their parameter states.

You can select fixtures by clicking the rows in the fixture list, or press command/ctrl + F and then type in the fixture ID (you can also specify a range by using - as thru).

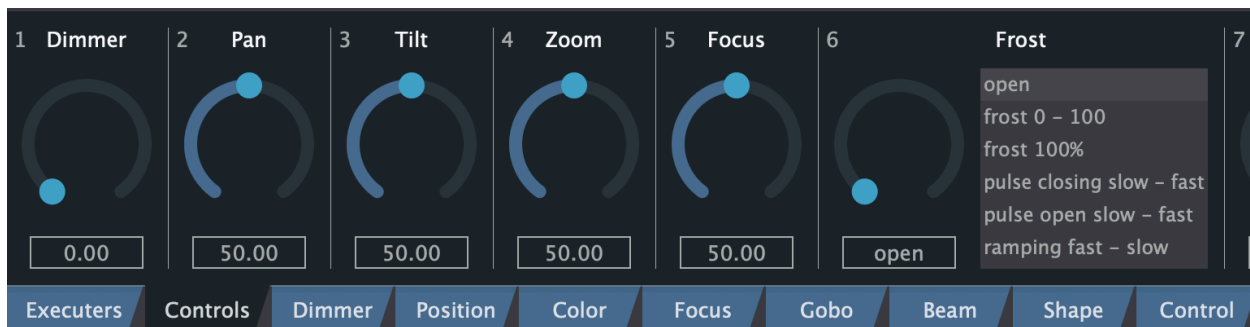
Fixture List						front	X
Select	ID	Name	Type	Dimmer	Position	Color	
	1	Apron 1	Dimmer	63.0			
	2	Apron 2	Dimmer	63.0			
	3	Apron 3	Dimmer	63.0			
	4	Apron 4	Dimmer	63.0			
	5	Apron 5	Dimmer	63.0			
X	6	DS 1	Dimmer	17.4			
	7	DS 2	Dimmer	0.0			
	8	DS 3	Dimmer	0.0			
	9	DS 4	Dimmer	0.0			
	10	DS 5	Dimmer	0.0			
	11	MS 1	Dimmer	0.0			
	12	MS 2	Dimmer	0.0			

You can filter the fixtures by their layer name, check the combo box on the top right corner.

After selecting some fixtures, you can change their parameters in the control tabs at the bottom. The parameter you changed will be displayed as red. This is called programmer value, it means it is not saved yet. If the parameter is coming from cuelist playback, it is yellow. Otherwise it is grey as default.

To clear the selection, press ESC key once. To clear the programmer values, press ESC again. All the red values will be lost.

## Controls



As mentioned above, the bottom control tabs show all the parameters that you can change with selected fixtures. The “controls” tab includes the quick access to the common parameter types. All parameters are in the following categorized tabs.

Each parameter is represented as a knob, if it has options defined in its profile, you will also see a list. You can click the number and type in the exact number (0-100) there, or use mouse to control the knob. Hold ctrl/command for fine control.

Right click on a parameter knob, you will find more advanced control options, such as setting the delay/fade timing, default and effect on that parameter.

## Group and Preset

Select fixtures in the fixture list or by fixture ID and change their parameters one by one is too much work sometime. To speed up the programming, group and preset are introduced.

**Group** stores a set of fixtures, it helps you speed up your fixture selection. For example, you can select all the beam fixtures and press an empty group cell. It stores not only the fixtures but also the orders you select in to the group. Next time when you want to re-select these fixtures, you can just press that group cell or press command/ctrl + G and then enter the group number. The group does not have to be mutual exclusive. But the selection are additive, that means if you keep re-calling different groups, all of them will be selected. If you just want to select one group, you need to clear your selection by press the ESC key first.

**Preset** is a way to store of set of parameter changes, you can later recall it back to the programmer. For example, you selected one RGB LED fixture and dialed in a purple color (RGB value) that you really like. You don't want to do turn those parameter knobs exactly like this one every time when you want that purple color. Then when the changes are still in the programmer, press an empty preset cell to store it.

You may notice the category drop box on the top right corner of the preset windows. Those are not just filters. They are different categories of presets. For example, color category will only store color parameters. (All will store everything changed).

The greatest advantage of using preset is later when you store it in a cue, it's still referenced by preset not hard values. That means, you can later update the preset (by right click on the cell) and then all your cues that refer to this preset will automatically be updated.

## Cuelist

OK, you have changed many parameters, they are in red and it looks pretty good. What now? You want to save it some where.

Introducing the most important concept in MaizeDMX, cue list. As the name suggested, it is a list of cues, and a cue stores the changes you just made. With the changes in the programmer, press an empty executor button, it will create a new cuelist with the current changes as the first cue, and assign it to the executor you clicked.



An executor is a set of controls to an object, it usually includes buttons and fader. The first tab at the bottom shows you a bank of executors (10). Click the header of an executor will make it the primary(selected) executor, which will be shown in a bigger view at the bottom right corner of the window.

To see the content of this cuelist, check the cue list content panel:

Cue List Content											
Selected											X
ID	Name	Trigger	Timing	MIB	Dimmer Ti...	Position Ti...	Color Timing	Focus Timing	Gobo Timing	Beam Timing	Sha
1	Cue 1	Go	0.0, 0.0, L		0.0, 0.0, L	0.0, 0.0, L	0.0, 0.0, L	0.0, 0.0, L	0.0, 0.0, L	0.0, 0.0, L	0.0
<div>Options</div> <div>Rename</div>											



Each cue has the following properties that you can config:

- ID: this has to be unique for each cue, and it determines the order of cues. You can append one decimal point to it, for example 1.1
- Name: right click on it to rename the cue.
- Trigger: Go or follow.
- Timing: Set the overall delay and fade time. Also the timing curve type.
- MIB: move in black. When enabled, it will move the position, color and gobo changes to previous cue when possible.
- Category Timing: set timing information on each parameter category.

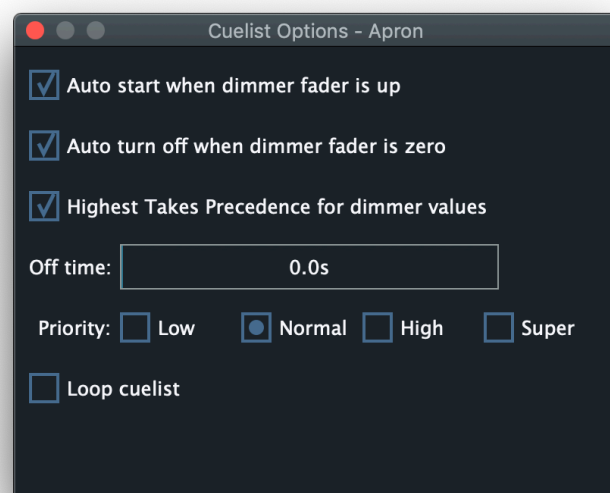
Press the space bar to go to next cue on the main executer, at this time you can clear the programmer by press the ESC key two times, as the parameter values are coming from cuelist playback now.

To create the next cue, change the parameters you need to change, and then press ctrl/command + R to record another cue in the main cuelist. Repeat this until you have all the cues that you need.

Tips: You can select cues and copy/paste them to other locations.

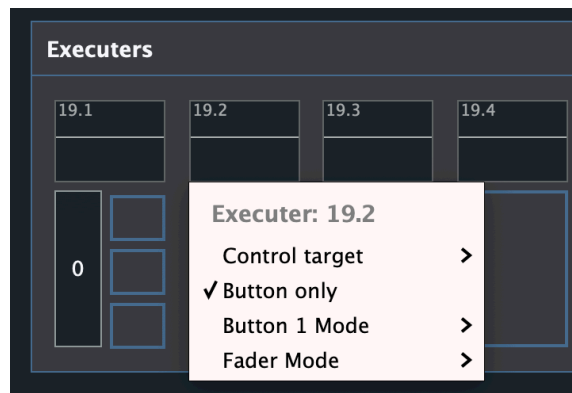
Click the “options” button at the bottom right, you will see the cruelest option windows:

- Auto start: If it should go to the first cue the dimmer fader of this cuelist is raised.
- Auto off: If it should turn off the cuelist if the dimmer fader is lowered to 0.
- HTP: When other cuelist is controlling the same dimmer values, highest dimmer value will win. Otherwise, the latest will win.
- Off time: How long does the dimmer value fade out when cuelist is turned off.
- Priority: When interacting with other plays, priorities are considered. High priority cuelist won't be turned off by turn off all executers command (ctrl/command + K). Super priority is even higher than programmer values.
- Loop: If cuelist should go back to first cue instead of turning off after the last cue.



# Executer

As we mentioned above, executer is a set of control (button and fader) to an object (usually cuelist).



The first tab at the bottom of the window is a quick way to access different banks of executer. You can also add executer panel into the workspace.

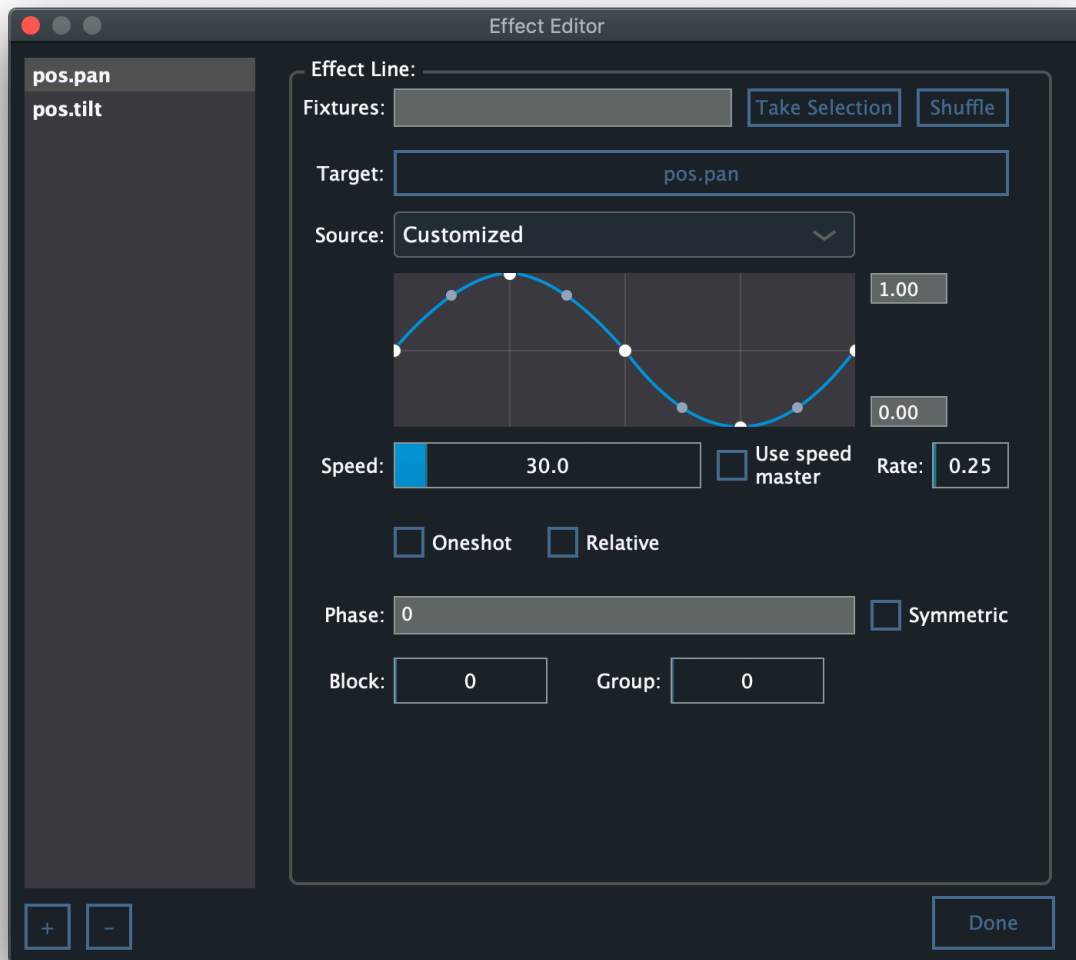
Right click on the header of an executer, you will see a popup. Executer can be changed to button only mode, which only has one button. Otherwise, it has a fader and three buttons. The action of the button and fader can be changed in the button mode and fader mode sub menu.

Now you know everything to create a cuelist for each program/piece you are designing. Assign them to executers. Select an executer as the main executer, and then just press space bar as things happen on stage. You can also create some generic single cue cuelist and assign them to executers, and then busking with faders and buttons.

# Effect

What if we want a parameter keep changing in a pattern instead of a fixed value? Effect engine is designed for that. There are two ways to add effect:

1. Temporary effect: After selecting the fixtures, right click on the control knob of the parameter than you want to add effect, then choose effect... in the popup menu. This adds effect just to that parameters.
2. Saved effect: In the effect panel in the workspace, click on an empty cell. This will show the effect editor with multiple layers. You can apply and reuse this saved effect on your selected fixtures.



In the effect editor, you can see a layer list on the left for saved effect. This means you can have an effect that control multiple parameters at the same time. Let me explain the details in effect editor:

- Fixtures: the fixture IDs that this effect line is designed to apply to. If empty, fixtures have to be selected before apply the effect. Otherwise, simply click on the effect cell will apply effect to these fixtures. The take selection button will copy the current selection here. The shuffle button will randomize the fixture order, as you will see this order is important when applying effect with different phases.
- Target: this is the parameter type that this effect is controlling.
- Source: the waveform that is controlling the parameter target, there is a list of presets that you choose. However, you can always tweak the wave form in the waveform editor as you like. On the right of the waveform, you can change its range from 0 to 1.

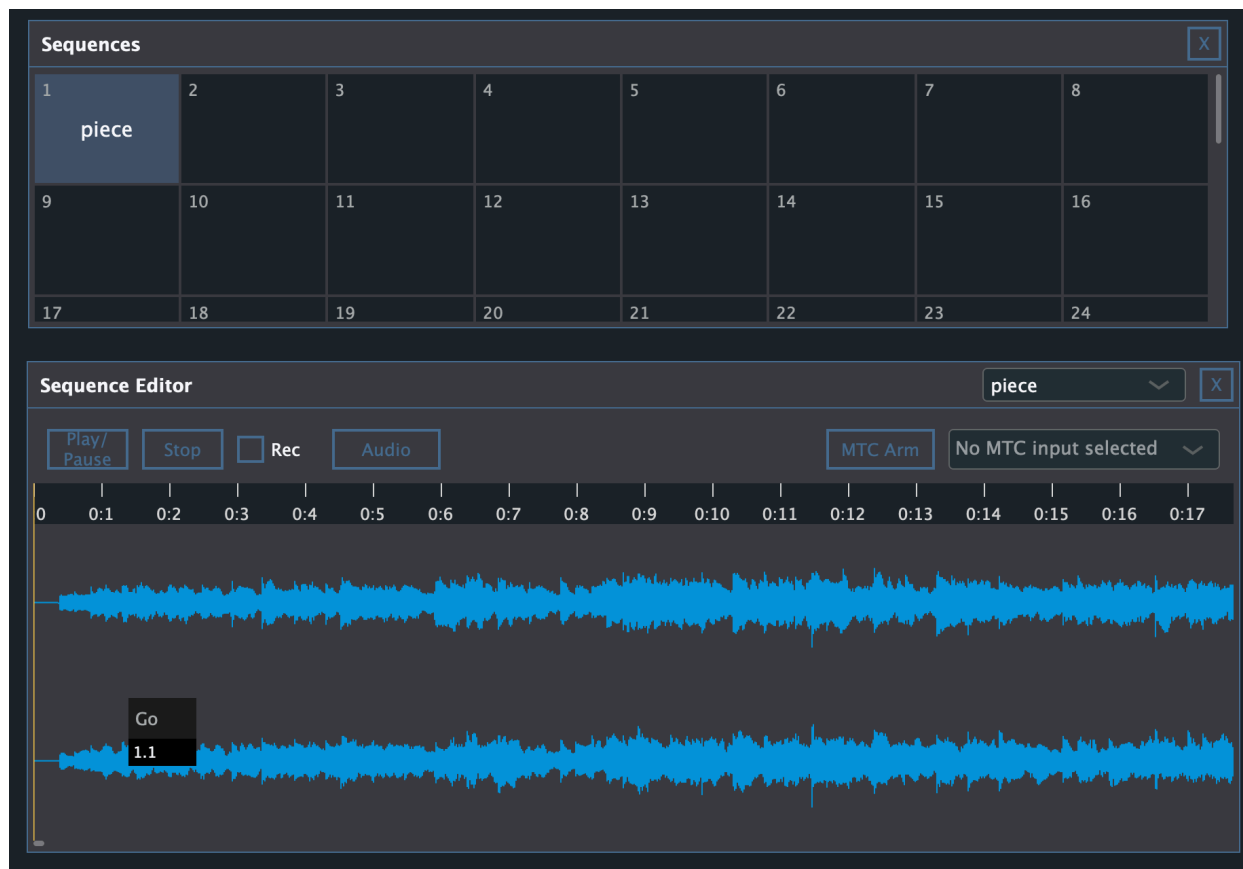
- Speed: how fast this effect run through the waveform. It is the BPM x Rate. If you check the use speed master, it will use the master BPM speed x Rate.
- Oneshot: if checked, the effect will just run through the waveform once, not looping.
- Relative: if checked, the effect will be based on the parameter's current value as the base. The 0.5 line in the middle of the waveform will represent the base.
- Phase: the starting phase of the effect, range from 0 to 1. It can also be range like "0-1" or "0-1-0", which means the fixtures will spread evenly between this range.
- Symmetric: if checked, the values will be flipped for the second half of the target fixtures.
- Block: this indicates how many fixtures in the target should behave as the same phase.
- Group: this indicates how many fixtures should be count as one phase cycle in the spread.

## Sequence

Cuelist is great, but what if you forget to press the space bar to go to next cue?

Synchronize lighting changes with music in sequence.

Add a panel of "sequences", this shows all your existing sequences. Click on an empty cell to create a new sequence. Then add the "sequence editor" panel, and select the new sequence you created from the top right combo box to edit/view this sequence.



Click the “Audio” button, assign an audio file to this sequence and then you can see the wave form of this file. Click play/pause or stop button to preview the music. Stop the music, and check the “rec” box and then press play button to start record events on the track. While the track is playing and the “rec” check box checked, operate on executors like you should. After it is done, press “Stop” button, you will see all the events recorded or tweak their timing if needed.



```
1 var midiInput = "iCON iControls";
2
3 function midiNoteOn(ch, note, velocity) {
4   var bank = ui.getCurrentPage();
5
6   if (note == 26) {
7     dmx.pressExecButton(1, ch + bank - 1, 1);
8   } else if (note == 27) {
9     dmx.pressExecButton(1, ch + bank - 1, 2);
10  } else if (note == 25) {
11    dmx.clear();
12  } else if (note == 24) {
13    ui.group(2);
14  } else if (note == 22) {
15    ui.group(3);
16  } else if (note == 21) {
17    ui.group(1);
18  }
19 }
20
21 function midiNoteOff(ch, note) {
22   var bank = ui.getCurrentPage();
23
24   if (note == 26) {
25     dmx.pressExecButton(1, ch + bank - 1, 1, false);
26   } else if (note == 27) {
27     dmx.pressExecButton(1, ch + bank - 1, 2, false);
28   }
29 }
30
31 function midiController(ch, type, value) {
32   var bank = ui.getCurrentPage();
33
34   if (type == 13) {
35     if (ch == 9) {
36       dmx.setMasterDimmer(value / 127.0);
37     } else {
38       dmx.setExecFader(1, ch + bank - 1, value / 127.0);
39     }
40   } else if (type == 12) {
41     if (ch < 9) {
```

Now during the show, you can just play this sequence and call it done. If you prefer the audio guy to play the music, MIDI time code can be set to drive this sequence. (select the MTC input from the combo box on the right).

Tips:

- Make sure you reset all the cuelists (ctrl/command + K) before you start recording or playback of a sequence. This makes sure you have a clean base to start with.
- The audio playback device can be picked in the general section of the settings tab.

## Controller

Use mouse and keyboard on executors can be slow. Do you have a MIDI controller with fader and buttons? If so, great! You can support it by writing some javascript :) In the controllers section of the settings tab, you can add any number of controller and write or paste the supporting javascript in the code editor below.

First declare the MIDI input and output name that it should look for as:

```
var midiInput = "xxxx";  
var midiOutput = "xxxx";
```

The callback functions that you can implement are:

function midiNoteOn(ch, note, velocity)	called when MIDI note on are received on midiInput
function midiNoteOff(ch, note)	called when MIDI note off are received on midiInput
function midiController(ch, type, value)	called when MIDI controller messages are received on midiInput
function currentPageChanged(bank)	called current page of executor (in the first bottom tab) is changed
function executorStatusChanged(bank, index)	called when an executor's status is changed

Some objects and functions are provided for you to call, they can control some parts of MaizeDMX:

midi.noteOn(ch, note, velocity)	send MIDI note on to midiOutput
midi.noteOff(ch, note)	send MIDI note off to midiOutput
midi.controller(ch, controller, value);	send MIDI controller message to midiOutput
ui.getCurrentPage()	get current page of quick executors (in the first bottom tab)

ui.getCurrentTime()	get system time as milliseconds
ui.increaseParameterValue(index, delta)	increase the #index parameter control on current control tab by delta amount
ui.decreaseParameterValue(index, delta)	decrease the #index parameter control on current control tab by delta amount
ui.releaseParameter(index)	release the #index parameter control value on current control tab
ui.pageUp()	increase page number of the quick executors
ui.pageDown()	decrease page number of the quick executors
ui.control(index)	switch control tabs
ui.workspace(index)	switch workspace tabs
dmx.clear()	just like you press ESC key
dmx.pressExecButton(bank, index, buttonIndex, buttonUp)	press one of the button of an executor, buttonUp is an optional boolean (true/false) to indicate this is a button down or up event.
dmx.setExecFader(bank, index, value)	set the fader value from 0 to 1 of an executor
dmx.setMasterDimmer(percent)	set master dimmer value from 0 to 1 in float number
dmx.toggleBlackOut()	black out button

## Appendix

---

### Fixture Profile Format

It's in JSON format, check out the some example profiles that you can download from our website. The majority of the description is about each parameter of the fixture, for example:

```
{
  "type": "control.control",    // All types are pre-defined strings
  "name": "switch",           // Display name in the control tab
  "channel": 1,                // This can be 16bit channel defined as "1, 2"
  "default": 0.5,              // default value if not 0
  "highlight": 1,              // value when the fixture is highlighted
  "options": {                 // With this option ranges, a list will be displayed next to the parameter knob
    "0-127": "off",
    "128-255": "on"
  }
}
```

## Parameter types (all of these are self explanatory):

```
const CParameterType kParameterType_Dimmer_Dimmer = "dimmer.dimmer";
const CParameterType kParameterType_Dimmer_BackgroundDimmer = "dimmer.backgroundddimmer";
const CParameterType kParameterType_Dimmer_Curve = "dimmer.curve";
const CParameterType kParameterType_Dimmer_Shutter = "dimmer.shutter";
const CParameterType kParameterType_Dimmer_Strobe = "dimmer.strobe";

// eParameterCategory_Position
const CParameterType kParameterType_Position_Pan = "pos.pan";
const CParameterType kParameterType_Position_Tilt = "pos.tilt";
const CParameterType kParameterType_Position_PanContinuous = "pos.pancontinuous";
const CParameterType kParameterType_Position_TiltContinuous = "pos.tiltcontinuous";
const CParameterType kParameterType_Position_Speed = "pos.speed";

// eParameterCategory_Color
const CParameterType kParameterType_Color_Wheel1 = "color.color1";
const CParameterType kParameterType_Color_Wheel2 = "color.color2";
const CParameterType kParameterType_Color_Wheel3 = "color.color3";
const CParameterType kParameterType_Color_CTO = "color.cto";
const CParameterType kParameterType_Color_Cyan = "color.cyan";
const CParameterType kParameterType_Color_Magenta = "color.magenta";
const CParameterType kParameterType_Color_Yellow = "color.yellow";
const CParameterType kParameterType_Color_Red = "color.red";
const CParameterType kParameterType_Color_Green = "color.green";
const CParameterType kParameterType_Color_Blue = "color.blue";
const CParameterType kParameterType_Color_Amber = "color.amber";
const CParameterType kParameterType_Color_White = "color.white";
const CParameterType kParameterType_Color_WarmWhite = "color.warmwhite";
const CParameterType kParameterType_Color_CoolWhite = "color.coolwhite";
const CParameterType kParameterType_Color_Orange = "color.orange";
const CParameterType kParameterType_Color_Lime = "color.lime";
const CParameterType kParameterType_Color_Indigo = "color.indigo";
const CParameterType kParameterType_Color_UV = "color.uv";
const CParameterType kParameterType_Color_Hue = "color.hue";
const CParameterType kParameterType_Color_Saturation = "color.saturation";

// eParameterCategory_Focus
const CParameterType kParameterType_Focus_Zoom = "focus.zoom";
const CParameterType kParameterType_Focus_ZoomRotation = "focus.zoomrotation";
const CParameterType kParameterType_Focus_Focus = "focus.focus";
const CParameterType kParameterType_Focus_AutoFocus = "focus.autofocus";

// eParameterCategory_Gobo
const CParameterType kParameterType_Gobo_Gobo1 = "gobo.gobo1";
const CParameterType kParameterType_Gobo_Gobo1Rotation = "gobo.gobo1rotation";
const CParameterType kParameterType_Gobo_Gobo2 = "gobo.gobo2";
const CParameterType kParameterType_Gobo_Gobo2Rotation = "gobo.gobo2rotation";
const CParameterType kParameterType_Gobo_Gobo3 = "gobo.gobo3";
const CParameterType kParameterType_Gobo_Gobo3Rotation = "gobo.gobo3rotation";
const CParameterType kParameterType_Gobo_Animation = "gobo.animation";
const CParameterType kParameterType_Gobo_AnimationRotation = "gobo.animationrotation";
const CParameterType kParameterType_Gobo_AnimationRotation2 = "gobo.animationrotation2";
const CParameterType kParameterType_Gobo_AnimationProgram = "gobo.animationprogram";

// eParameterCategory_Beam
const CParameterType kParameterType_Beam_Frost = "beam.frost";
const CParameterType kParameterType_Beam_Iris = "beam.iris";
const CParameterType kParameterType_Beam_Prism1 = "beam.prism1";
const CParameterType kParameterType_Beam_Prism1Rotation = "beam.prism1rotation";
const CParameterType kParameterType_Beam_Prism2 = "beam.prism2";
const CParameterType kParameterType_Beam_Prism2Rotation = "beam.prism2rotation";

// eParameterCategory_Shape
```



```
const CParameterType kParameterType_Shape_Preset = "shape.preset";
const CParameterType kParameterType_Shape_PresetSpeed = "shape.presetspeed";
const CParameterType kParameterType_Shape_PresetFade = "shape.presetfade";
const CParameterType kParameterType_Shape_FrameAngle = "shape.frameangle";
const CParameterType kParameterType_Shape_Blade1 = "shape.blade1";
const CParameterType kParameterType_Shape_Blade1Angle = "shape.blade1angle";
const CParameterType kParameterType_Shape_Blade2 = "shape.blade2";
const CParameterType kParameterType_Shape_Blade2Angle = "shape.blade2angle";
const CParameterType kParameterType_Shape_Blade3 = "shape.blade3";
const CParameterType kParameterType_Shape_Blade3Angle = "shape.blade3angle";
const CParameterType kParameterType_Shape_Blade4 = "shape.blade4";
const CParameterType kParameterType_Shape_Blade4Angle = "shape.blade4angle";

// eParameterCategory_Control
const CParameterType kParameterType_Control_Control = "control.control";
const CParameterType kParameterType_Control_Reset = "control.reset";
const CParameterType kParameterType_Control_Program = "control.program";
const CParameterType kParameterType_Control_ProgramSpeed = "control.programspeed";
const CParameterType kParameterType_Control_ProgramFade = "control.programfade";
const CParameterType kParameterType_Control_Haze = "control.haze";
const CParameterType kParameterType_Control_Fan = "control.fan";
```