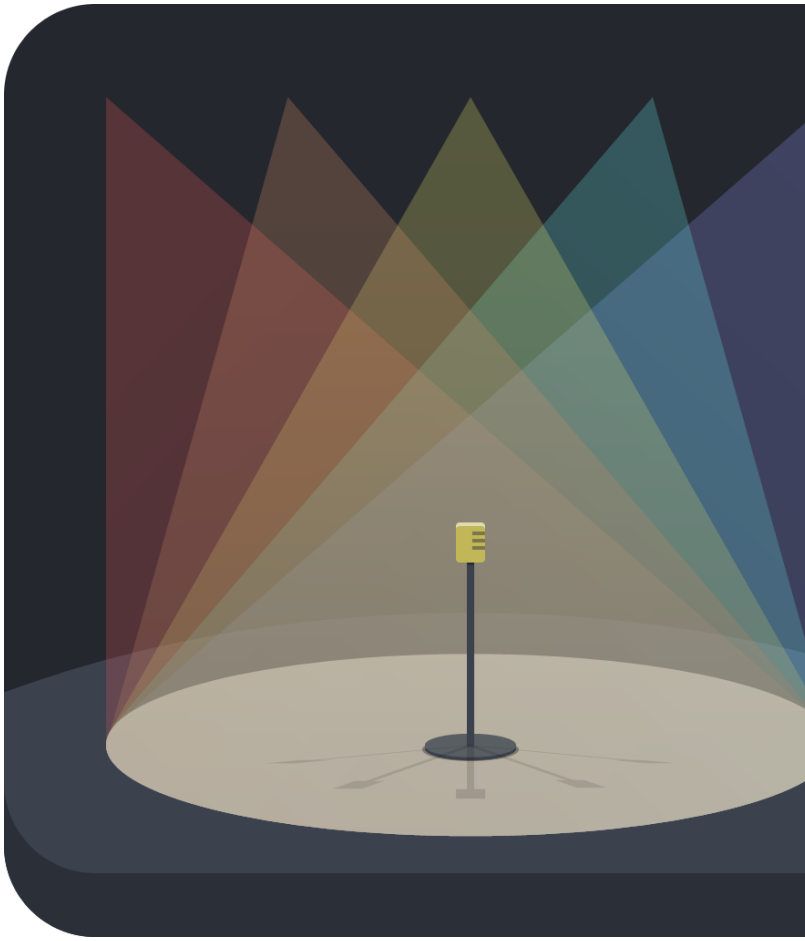
MaizeDMX 3

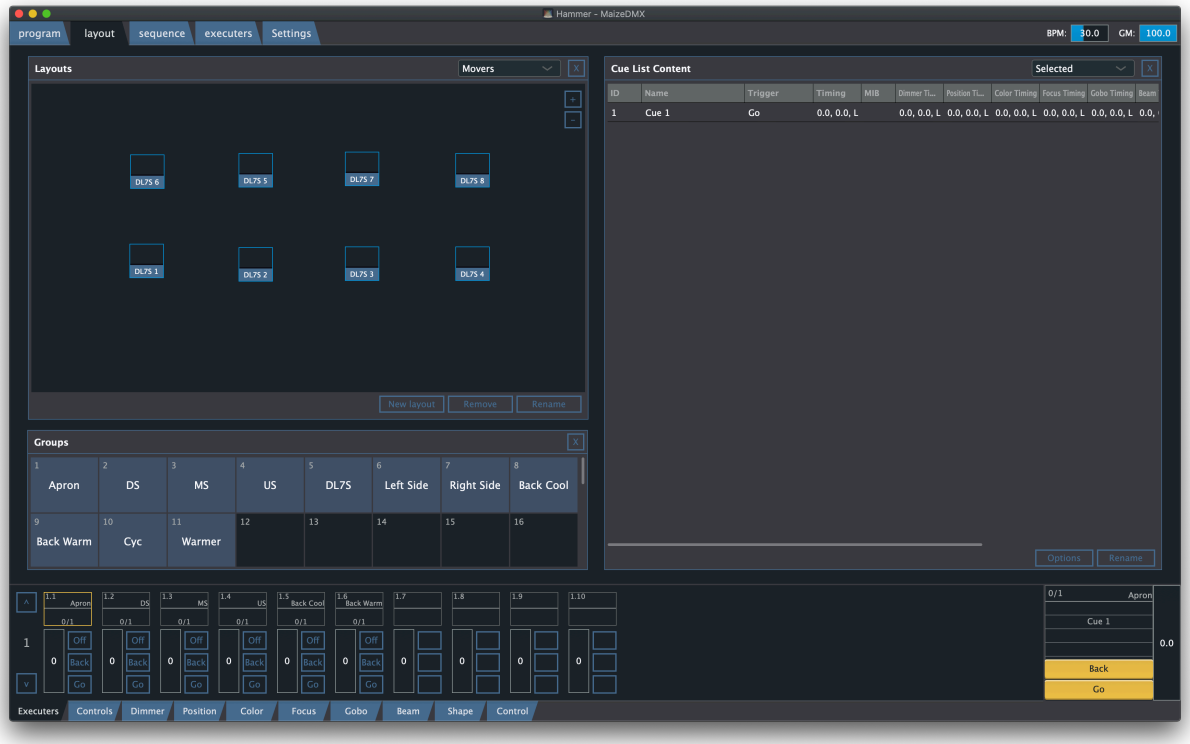
用户手册



MaizeDMX是一个简单易用的灯光控制软件。如果你是一个刚入门的灯光设计师或操作员，你会发现这个软件非常适合您的演出和活动。

- 支持基于网络和USB的DMX接口。
- 跨平台支持Windows和MacOS操作系统。
- 标准化的灯控概念，比如编程器，编组，素材，Cuelist，效果等。
- 开放的JSON格式灯控文件格式。
- 用时间序列同步音乐和灯光。
- 通过Javascript支持各种MIDI控制器。

总览



MaizeDMX 3的主窗口分为上下两部分，上部分为工作区域，下部分为控制区域。

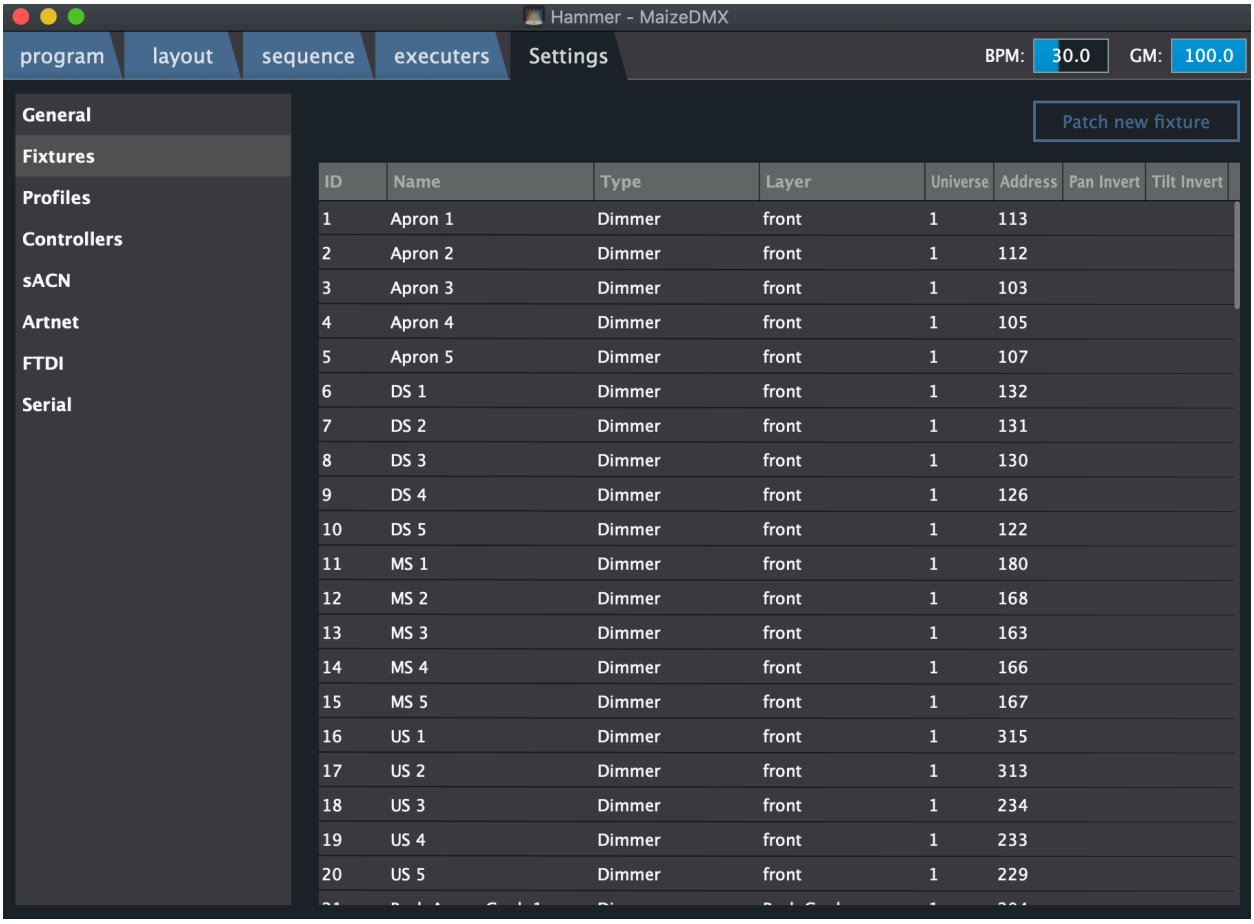
你可以在上面的选项卡里建立自己的工作区域，在工作区域空白的地方按下鼠标右键，然后可以添加各种功能板块。每个功能板块都可以随意被放置和放大缩小。(可以用F1-F8快捷键切换当前显示的工作区域)

下方的控制区显示快捷回放执行器(Executers)页面，当你选中灯具后，也可以在后面的选项卡中控制其各种参数。(可以用Alt + F1-F8来切换控制页面)

现在想象一下要做一台演出的灯光，让我们来一步一步了解MaizeDMX的各种基本功能。

配置 (Settings)

工作区域的最后一个页面总是配置页面，这是我们要做的第一步。

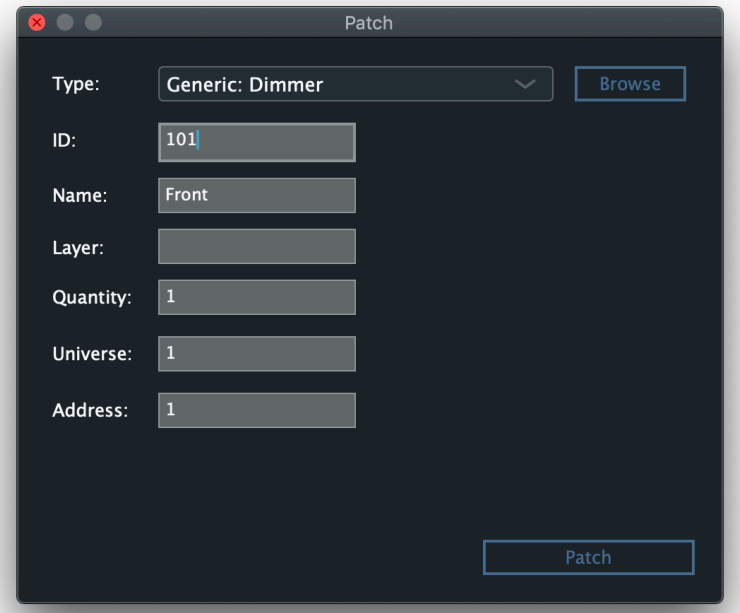


输出(Output)

要真的控制灯具，总得有个办法输出DMX信号。电脑本身并没有DMX接口，但是我们支持以下几种输出设备：

- 基于USB的DMX转接器：MaizeDMX支持例如Enttec和DMXking的USB DMX设备。你应该在左边的FTDI或Serial区域能找到他们。如果没有的话，你可能需要安装驱动程序 here.
- 基于以太网络的DMX节点: 你可以在sACN或Artnet区域找到他们的配置。注意要选择Enable启动某种输出方式。

配接灯具(Fixtures)



在当前工程中增加灯具，点左边“Fixtures”页面, 然后在右面点击右上角有一个“Patch new fixture”按钮.

在第一个下拉列表中选择灯具类型，估计你会发现只有一种就是传统灯具(Dimmer)。你需要点击Browse按钮浏览找到你想要灯具的描述文件(*.json)。你可以在我们的网站下载一个打包文件，里面有一些我们制作的灯库。但是很显然，大部分灯具是没有的。如果你有一个特别的灯具需要添加，你需要自己写这个灯具的描述文件。不要害怕，没那么困难，我们在后面会介绍。

第二个文字框是ID，这是一个灯具的唯一数字标识。你在编程的时候可以通过这些数字来选择灯具。

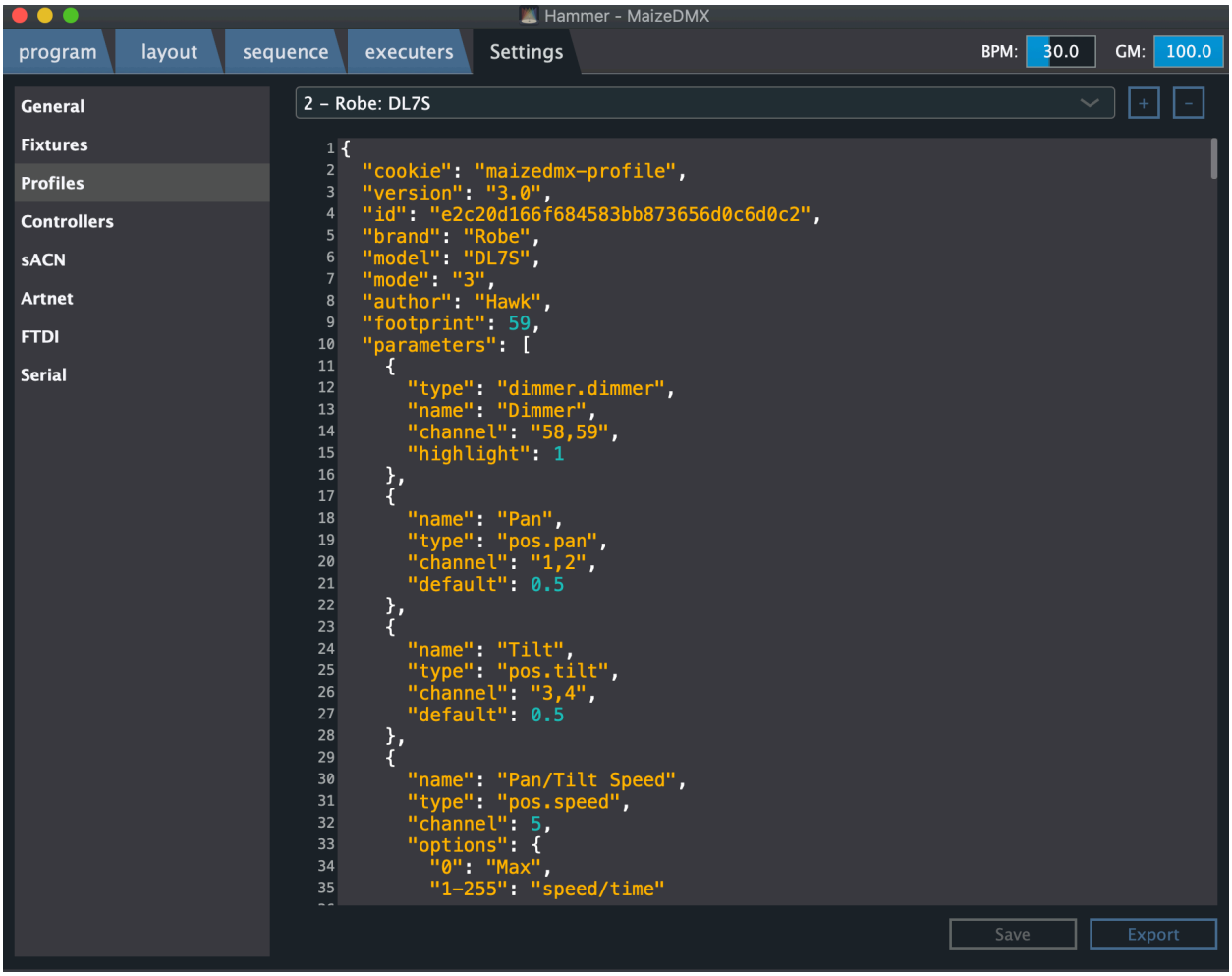
第三行layer是把灯具自定义分类的办法，不写也可以。

第四行Quantity是要配接此类灯具的个数。如果超过一个，MaizeDMX会自动分配连续的ID和地址。

最后两行就是地址了，Universe和address一定要填正确。

最后点击右下角的patch按钮，你就会看到在灯具列表里新加入的灯具。你可以右键点击然后修改某个灯具的信息，甚至可以多选，然后修改。

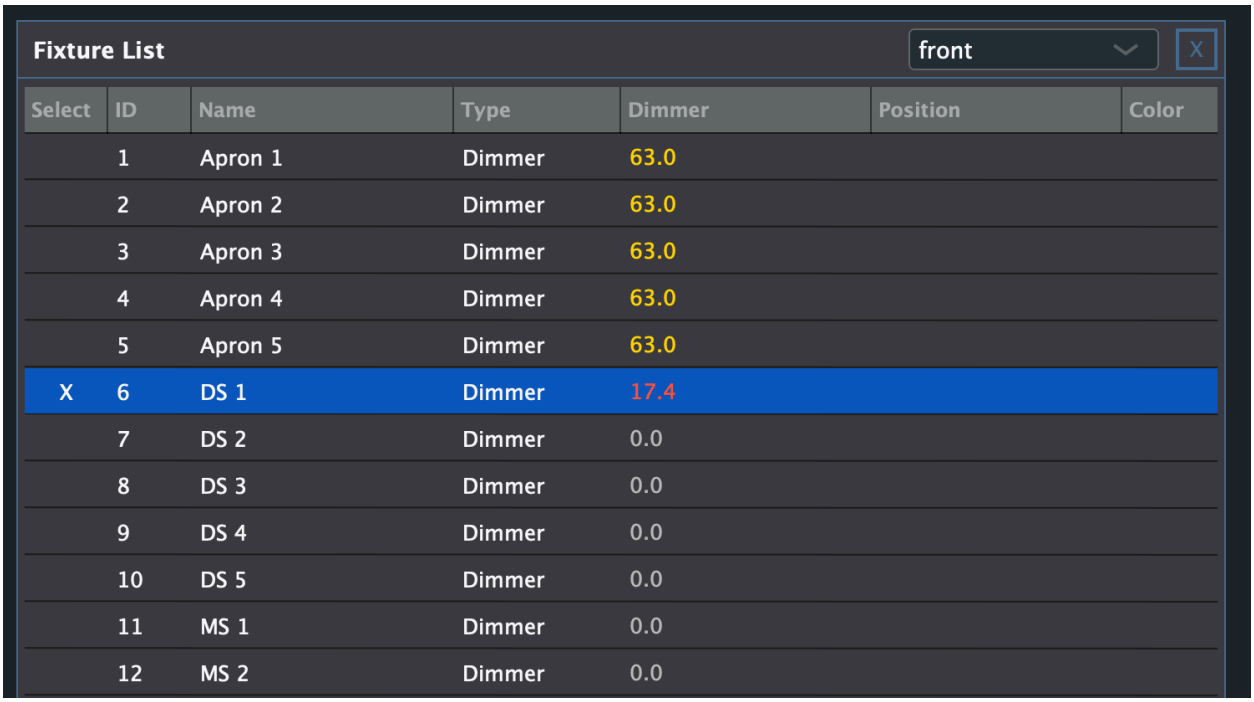
灯具描述 (Profiles)



刚说过你可能得自己写灯库文件的问题，在Profiles页面里，可以按+按钮新建灯具类型，在下面的代码输入区写JSON。但是我们建议还是使用外面的文本编辑器写好存下，然后在配接灯具时直接找到这个文件，这样以后其他工程还可以重复使用。文件的格式其实很简单，看上面的截图，最重要的就是要告知footprint(占用DMX地址空间大小)，然后描述每一个参数的类型。具体请看附录。

灯具列表(Fixture List)

我们现在切回第一个工作区域(F1)。 你应该能看到一个灯具列表板块，这里显示了你所有灯具的状态。



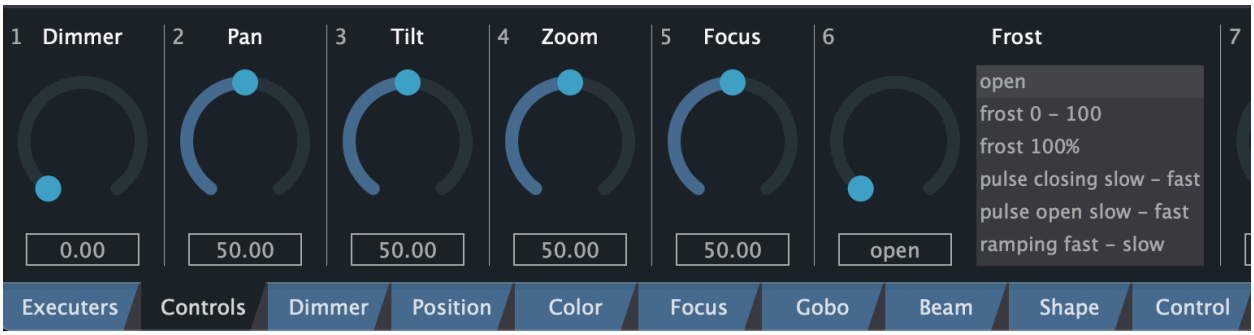
你可以用鼠标点击选择灯具，也可以按command/ctrl + F然后输入灯具ID选择。 (可以使用-表示几号到几号)

如果你当时输入了分层信息(layer)，你也可以用右上角的下拉列表来筛选你的灯具。

当选择了灯具以后，你就可以在下方的控制区域改变他们的各种参数了。你改变的参数将会显示成红色，这代表着这些数据还没有被保存。如果灯具参数来自于回放，则颜色是黄色的。如果是默认值没有改变，则是灰色的。

按ESC键一次，可以取消选择。再按一次，则会清除所有的红色临时数据。

控制区域(Controls)



我们刚才提到了，在主窗口下部的选项卡提供了所有可以修改的灯具参数。第二个页面是常用参数，后面是分门别类的参数。

每一个参数被标示为一个旋钮，如果灯控文件描述了这个参数的选项，则会看到了一个列表。你可以在下面的数据框里直接输入0-100的数字，或者用鼠标上下拖动旋钮。按住ctrl/command可以细调。

右键点击旋钮，你可以看到一些高级选项，比如说时间(延迟，渐入)，默认值和效果。

编组(Group)和素材(Preset)

是不是觉得每次选择灯具然后再在下面调一堆旋钮太麻烦了？为了加快编程的速度，编组和素材的概念产生了。

编组是一种快速选择灯具的办法。比如你可以选择所有的光束灯，然后点击一个空的编组单元格，这样就把这组选择存下了。下次，你只需要按这个单元格，或者按command/ctrl + G键，然后输入编组号，就可以再次选择这些灯具。编组不一定得是互斥的，但是选择是可以叠加的，这意思就是如果你不断按不同的编组，所有里面存的灯具都会被选中。想取消当前选择，你需要按一下ESC键。

素材是一种存贮参数变化值的方法，方便以后调用回编程器。比如说你有一种RGB调色灯，你终于通过调整RGB旋钮找到了你最喜欢的一种紫色。如果每次都要准确的跳到这几个RGB值的话就太麻烦了。这个时候你就可以点击一个空的preset单元格。

素材窗口右上角的下拉列表是不同类型的素材，也就是说如果你在Color类别存储preset，它只会记住颜色相关的参数。

其实素材最大的优势在于以后你把你的改变存到cue里以后，它存的是一个对素材的引用，而不是固定死的参数值。将来如果你对这个颜色不满意，只需要更新这个素材即可(右键点击这个素材单元格)。所有用到这个素材的Cue都会自动发生变化。

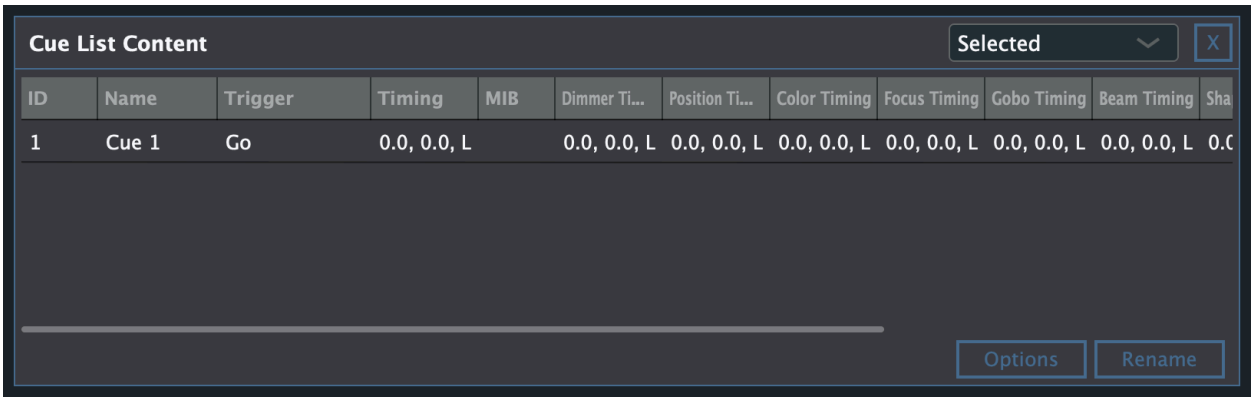
Cuelist

好，现在你已经修改了很多参数，舞台上已经很好看了。然后呢？当然是把他们存起来。

Cuelist是MaizeDMX里最重要的概念，它是一个或多个cue(变化)的序列，每个变化里存着你的所做的参数改变。在有红色编程器值的情况下，按任何一个空的执行器(executer)按钮，MaizeDMX就给你建立了一个新的Cuelist，第一个cue就是你刚刚做的改变。

执行器指的是一组对一个对象的控制方法，一般就是一套按钮和推子。控制区第一页就是所有的执行控制器，10个一页。鼠标左键点击执行器的表头，可以把它变成当前主执行器，他会被放大显示在窗口右下角。

如果想看一个cuelist里面的具体信息，请用Cue List Content功能板块：



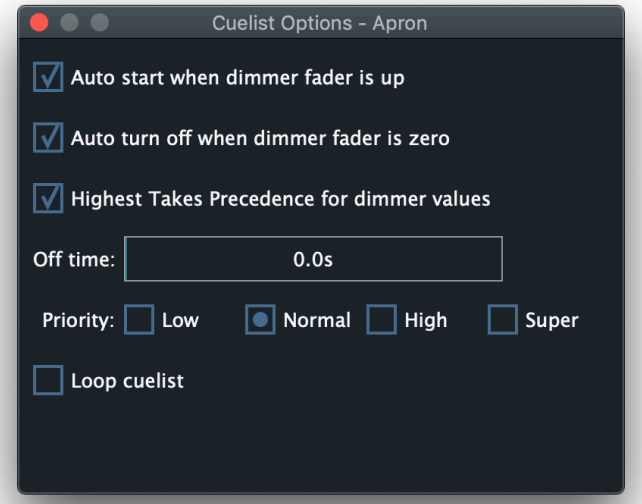
每一个Cue有以下属性:

- ID: 这是一个唯一标识数字，它决定了Cue的顺序。支持小数，如1.1的形式
- Name: 右键点击改名字
- Trigger: 激发方式
- Timing: 各种时间信息，包括过度时间，延迟还有时间曲线类型
- MIB: move in black. 如果勾选，会把一些会看到变动过程的参数移到前面的cue，比如色片，形状片等
- Category Timing: 每个类别的参数都可以单独设置时间信息

按空格键就相当于按主执行器的GO按钮，正式进入这个cuelist的第一个cue。这个时候你就可以按两次ESC键，清楚编程器的红色临时值。这样现在的状态就来自于回放了。

再次选择一些灯具，做一些参数改动，然后按CTRL/Command + R键，这样就建立了第二个Cue。重复这个过程直到你需要的所有Cue都完成。

小贴士: 你可以选择cue然后用ctrl c + ctrl v复制粘添到其他地方。



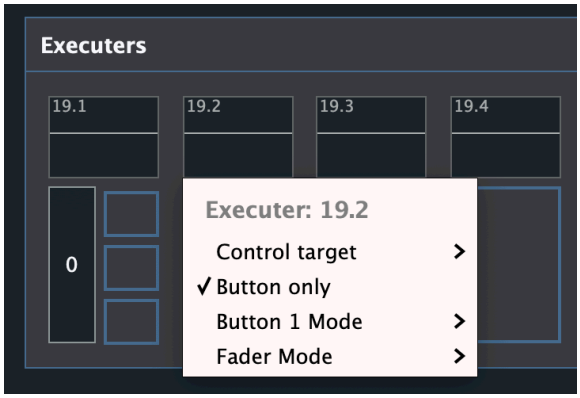
按右下角的“options”按钮, 你就可以看到Cuelist的一些选项:

- Auto start: 自动开始，是否当这个cuelist的dimmer推杆推起来的时候自动进入第一个cue。
- Auto off: 自动关闭，是否当dimmer推杆到0的时候自动关闭这个cuelist。
- HTP: 当和其他Cuelist共同作用的时候，是否按照最高的亮度值选择，否则就按照时间先后顺序了。
- Off time: 关闭时间，当关闭这个cuelist的时候用多长时间减弱。
- Priority: 当和其他Cuelist共同作用的时候，谁有更高的优先级。High priority高优先级的cuelist不会被ctrl/command + K关掉。Super priority超高优先级的甚至不会被编程器覆盖。
- Loop: 当最后一个cue执行完后是否循环到第一个。

执行器(Executer)

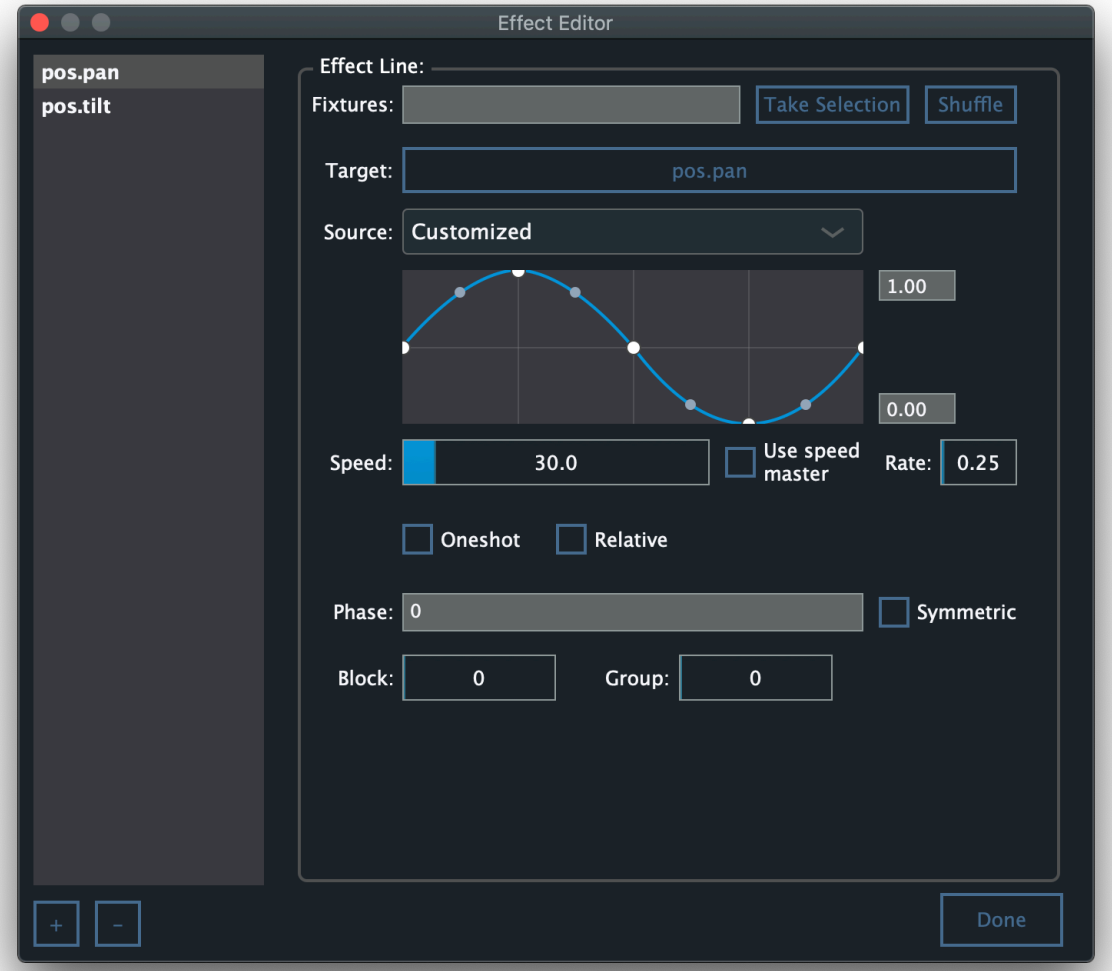
执行器是一套控制cuelist的组件。当然也可以控制别的东西。

控制区域的第一个页面有一系列执行器，也可以在工作区域添加任意多个执行器功能板块。右键点击执行器的表头，会出现一个快捷弹出菜单。你可以把这个执行器设置为Button only mode，即只有一个按钮。否则默认状态下，它有三个按钮和一个推子。具体推子和按钮做什么也可以在这个菜单中自定义。



现在已经了解了所有基本知识了，你应该掌握了怎么创建一个新的cuelist，并一步一步完成所有cue的制作，然后在演出时按空格键执行。你也可以再多创建一些只有一个cue的cuelist，然后在现场即兴发挥组合这些cuelist。

效果(Effect)



如果你想让你个参数不停地按照一个模式变化，而不只是一个固定的值，怎么办？效果器就是来干这个的。基本上有两种方法可以加效果：

1. 临时效果: 选择灯具后，在需要加效果的参数旋钮上按右键，然后选择效果…。这是针对你选择的灯具的这个参数加的效果。
2. 保存效果: 在工作区域可以添加一个效果板块，然后点击一个空的单元格。这样可以创建多层的可复用的效果。

在效果编辑器里，你可以在左边看到一列效果层，每层控制一个参数。下面介绍一下细节：

- Fixtures: 这里写着效果控制的目标灯具ID。如果不填，在使用效果时必须先选择灯具。如果填了，就可以直接单击效果使用。take selection按钮可以把当前选择的灯具放到这里。shuffle按钮可以随机打散排列一次，因为这个灯具顺序还是很重要的。
- Target: 效果控制的目标参数类型。
- Source: 控制曲线，在下拉列表里有一些预制好的曲线，当然你也可以在下面的图形里随便调节。图形右边是一个0-1的控制范围。0.5是中间。
- Speed: 效果运行速度。如果选择了use master speed，则会使用总的全局速度。
- Oneshot: 是否只运行一次，而不循环。
- Relative: 是否变成相对效果，意思是针对灯具此参数的当前值调整参数，而不是绝对值。0.5中线初意味着不改变。
- Phase: 效果的相位，0到1之间的一个数字。也可以是区间，类似0-1或0-1-0，这代表这灯具将会在这个区间里平均分布。
- Symmetric: 对称，对另一半的灯具反作用。
- Block: 多少个灯具作用相位相同。
- Group: 多少个灯具一个相位循环。

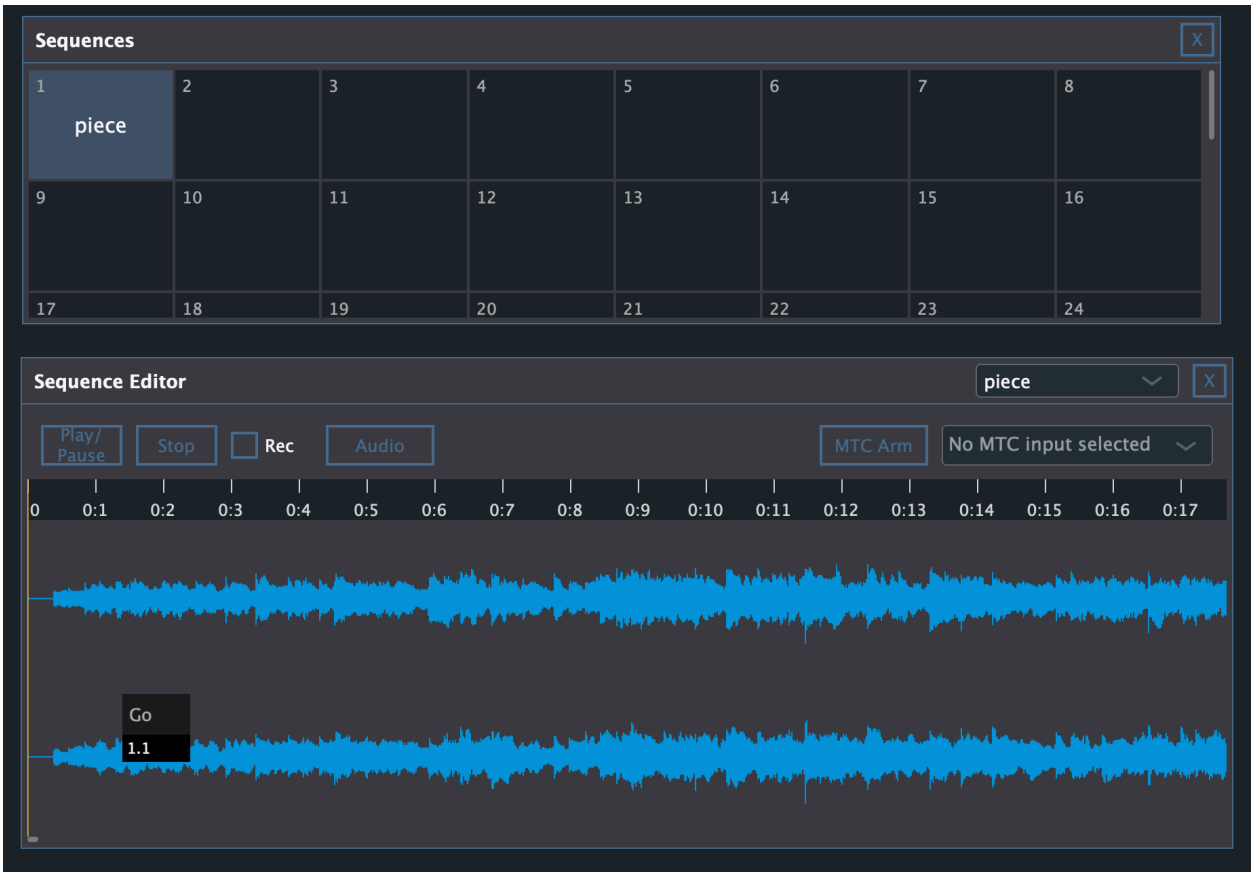
时间序列(Sequence)

Cuelist非常好，但是如果你忘了按空格键怎么办？用时时间序列可以把音频和灯光同步。

在工作区域添加一个“sequences”板块, 这个网格显示了所有已有的序列。点击一个空的单元格，新建一个序列。这时候你要再创建一个“sequence editor”板块，然后在右上角选择这个新序列来编辑和查看这个序列里的内容。

如下图，点击“Audio”按钮, 选择一个音频文件，如果读取没有问题，你就可以看到它的波形了。按“play/pause”按钮来预览一下。点击”Stop”按钮停止音乐，然后选中“rec”然后

再播放，这时就是录制模式，当你在操作各种executer时, 就可以看到事件被记录在了波形上面。停止后，你可以选中事件，对时间作调整。



有了时间序列以后，在演出时你就可以直接播放这个，然后什么都不用管了。如果你想让音响那边播放声音的话，你需要让他们在播放同样的音频文件时给你个MIDI时间码(MTC, 在右上角的下拉列表框里选择MIDI输入接口来接入时间码，准备好了以后别忘了按下Arm按钮，整个序列就由他们来驱动了，你这里不出声音)

小贴士:

- 在开始录制和播放之前，最好停止复位所有cuelist(按ctrl/command + K)，这样保证有一个干净的基础状态。
- 回放的音频设备可以在settings页面里选择。

控制器(Controller)

问题来了，用鼠标键盘来控制执行器也太慢了吧！你有MIDI控制器吗？如果有的话，你可以使用任何MIDI控制器来控制大部分MaizeDMX功能。但是你可能得写一点Javascript :)

在Settings页面的Controller区域，你可以添加控制器支持代码。下面介绍一下函数支持:

首先声明你的控制器的MIDI输入输出名称，这是在系统里显示的名称，你的控制器可能不需要输出。

```
var midiInput = “xxxx”;
var midiOutput = “xxxx”;
```

支持的回调函数如下:

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 executer (in the first bottom tab) is changed
function executerStatusChanged(bank, index)	called when an executer's status is changed

我们也提供了很多对象和函数让你来控制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 executers (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 executers
ui.pageDown()	decrease page number of the quick executers
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 executer, 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 executer

dmx.setMasterDimmer(percent)	set master dimmer value from 0 to 1 in float number
dmx.toggleBlackOut()	black out button

附录

灯控描述文件格式

如果你稍微懂点计算机知识的话，我们的文件格式是很常用的JSON, 你可以从我们的网站下载到一些示例灯控文件。如果你用文本编辑器打开，你会发现主要的部分就是参数的描述，比如说:

```
{
    "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"
    }
}
```

所有参数类型 (看名字就因该知道是什么意思了):

```
const CParameterType kParameterType_Dimmer_Dimmer = "dimmer.dimmer";
const CParameterType kParameterType_Dimmer_BackgroundDimmer = "dimmer.backgrounddimmer";
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";
```