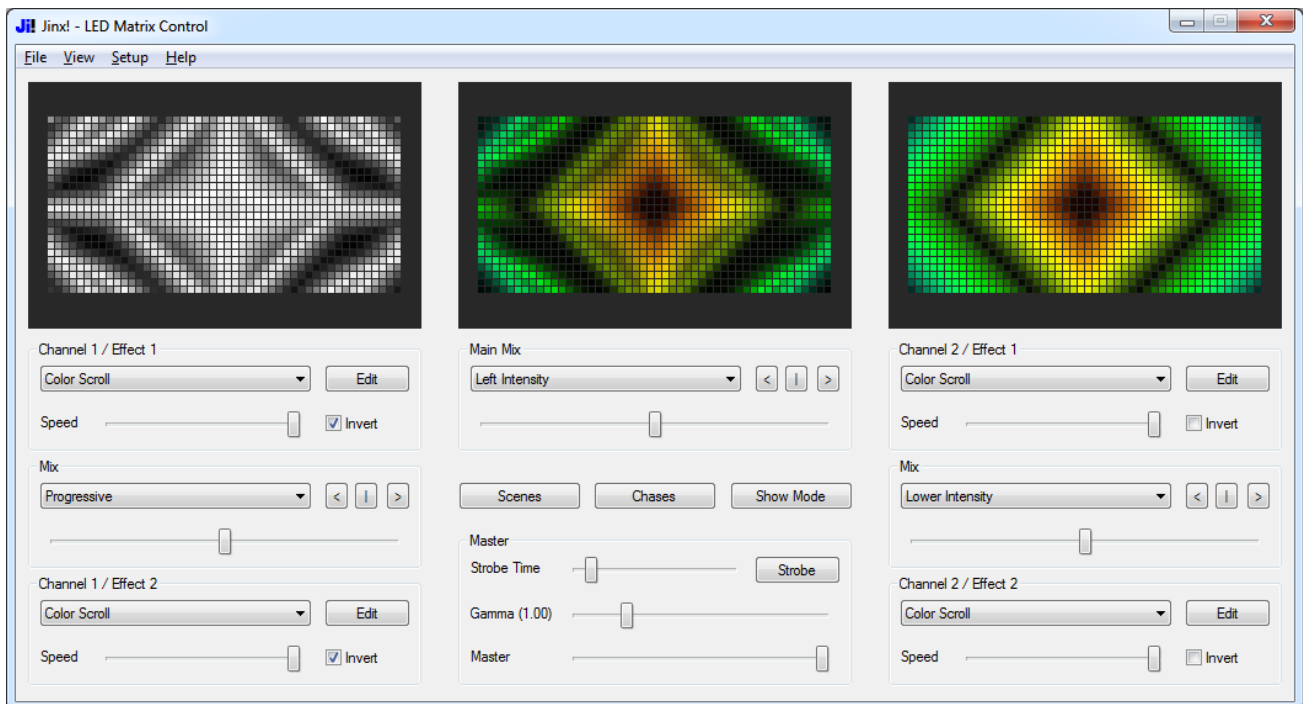


# Jinx! – LED Matrix Control



## User Manual

Version 1.5

© 2013-2014 Sven Karschewski  
<http://www.live-leds.de>

# Table of Contents

Features .....	4
Quick start .....	5
Matrix Options.....	5
Output Devices .....	5
Patch Matrix .....	6
Start Output.....	7
Main Window .....	8
Effect Generators .....	8
Copy and Paste Effects .....	9
Mixing Effects .....	9
Channel and Main Preview.....	10
Mixing Channels .....	10
Controlling Master Output .....	10
Main Window Buttons.....	11
Working with Scenes .....	11
Manage and Playing Scenes .....	12
Scene Fade.....	12
Working with Chases .....	13
Creating a new Chase .....	14
Starting and Stopping a Chase.....	16
Manage Chases.....	16
The Jinx! File Recorder.....	16
The Show Mode.....	17
Setup Matrix Size and Options .....	18
Configure Output Devices .....	19
Add and Edit Output Devices.....	19
Patch Matrix .....	22
Fast Patch .....	23
Starting Output.....	24
Remote Control .....	24
Audio based Effects .....	26
Auto Gain Control.....	26
Audio Trigger Setup.....	26

The DVI Output Window .....	27
Saving and Loading your Show .....	28
Importing a Show .....	28
Auto save and load .....	28
Command Line Options .....	29
Loading Files over the Command line .....	29
Auto start a scene.....	29
Auto start a chase.....	29
Invoke the show mode at Startup .....	29
Limit output frame rate to 20fps.....	29
Keyboard Shortcuts .....	30
The Jinx! Effect Engines .....	31
Simple Color.....	31
Color Scroll.....	31
Plasma .....	31
Fire.....	31
Metaballs.....	32
Expanding Shapes.....	32
Falling Rain.....	33
Radar/Scan Lines .....	33
Scrolling Text .....	33
Image Viewer.....	34
Starfield .....	34
Fading Pixels .....	35
Simple Lines.....	35
Sine Lines.....	35
Strobe .....	35
Spectrum Analyzer.....	36
Audio VU Meter.....	37
AVI Player .....	38
Capture Screen .....	39
Capture Webcam.....	39
Jinx! File Player .....	40
Jinx!Script .....	40

Jinx!Script Reference .....	41
Introduction.....	41
Variables .....	42
Arrays.....	42
Expressions and Math functions .....	42
Relational and Logical Operators .....	43
Labels, Subroutines and Jumps .....	43
If / Else Conditions.....	44
While / Wend Loop.....	45
For / Next Loop.....	45
Controlling Loops with Break / Continue .....	46
The Pset command .....	46
The Pget Command .....	47
The Line Command.....	47
The Rect Command .....	47
The Circle Command .....	47
The Text command.....	48
The Clear Command .....	48
The Fade command .....	48
The Config Command .....	49
System Variables .....	49
Comments inside the Source Code .....	49
Additional Coding Rules and Notes .....	50
Jinx!Script Command Chart .....	51

## Features

Jinx! is a standalone controlling software for led matrices, it has a powerful set of features to get the most out of your matrix.

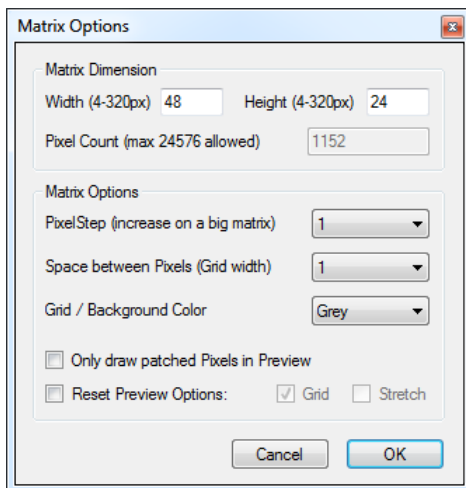
- simple, intuitive and fully resizable user interface
- 2 effect channels with 2 independent generators each
- 22 effect generators with lots of variations and sub effects including audio based effects like strobe and spectrum analyzer as well as image viewer with animated gif support, avi player and webcam support
- Create your own effects with Jinx!Script – a small but powerful basic like programming language to write effects on your own
- switchable auto gain control for audio capturing
- generators can be merged with a lot of different modes
- both effect channels can be merged and crossfaded
- simple master control for main brightness and gamma
- additional master strobe effect
- scene store to quickly save and access a composed effect
- powerful chase engine to build your own chases with stored scenes
- touchscreen friendly show mode for playing scenes and chases
- all preview panels configurable, channel preview panels switchable to single generators
- copy and paste single generators to another channel or scene
- record effects to a file which can be played in a single effect generator and combined with other effects to get an endless number of effect combinations
- various and flexible output options, supporting Art-Net, sACN (E1.31), tpm2.net, tpm2, Glediator protocol and MiniDMX as well as real DMX ( Enttec OpenDMX USB and Enttec USB DMX Pro compatible interfaces)
- switchable borderless DVI output window, which will stay on top and wipes out the mouse cursor
- all output protocols can be redirected to a file to create animation files for standalone controllers
- additional Bitmap Export to save the created animation as series of Windows Bitmap files
- create Glediator 2 / Solderlab UIB compatible recorder files
- multiple output devices can be handled to drive a serial matrix with more than one port
- flexible pixel to output device patching
- remote control support via DMX (Enttec USB DMX Pro compatible interfaces), Art-Net, sACN (E1.31), tpm2.net and tpm2 to control scene and chase changes, master brightness and master strobe remotely over a hardware or software based light desk
- complete ArtPoll support for Art-Net input and output
- complete user manual integrated as context sensitive help
- auto load show files and auto start scenes/chases and invoke the show mode via command line
- small, fast and native windows application, doesn't need any additional runtimes

## Quick start

If you can't wait to see something on your LED matrix, here are the basic steps to configure your matrix.

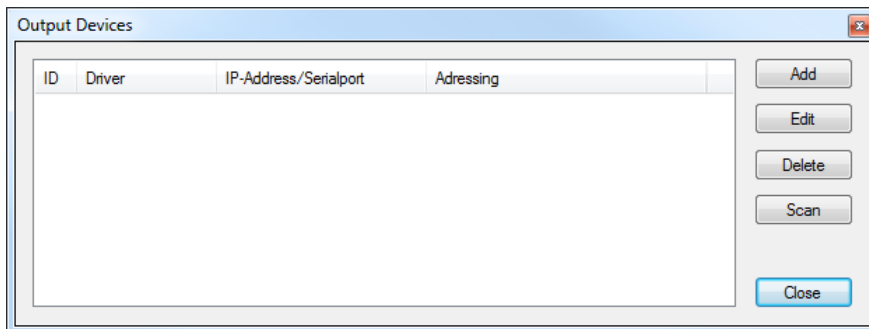
### Matrix Options

In the first step you should define your matrix size. To do this simply go to **Setup -> Matrix Options** and tell Jinx! your matrix dimensions.



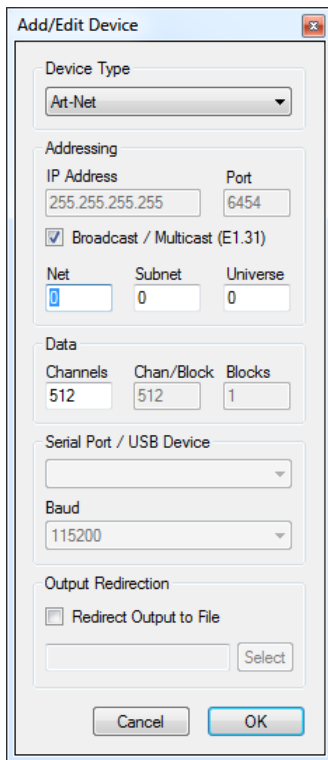
### Output Devices

After setting your matrix resolution, you should define your output devices. This will be found in the **Setup -> Output Devices** dialog.



## Jinx! – LED Matrix Control

Here you can manage your devices and even scan for Art-Net devices inside your network. Simply add one or multiple devices and give it the correct parameters.



The 'Add/Edit Device' dialog box is used for configuring device parameters. It includes sections for Device Type, Addressing, Data, Serial Port / USB Device, and Output Redirection.

**Device Type:** Art-Net

**Addressing:**  
IP Address: 255.255.255.255  
Port: 6454  
 Broadcast / Multicast (E1.31)  
Net: 0, Subnet: 0, Universe: 0

**Data:**  
Channels: 512, Chan/Block: 512, Blocks: 1

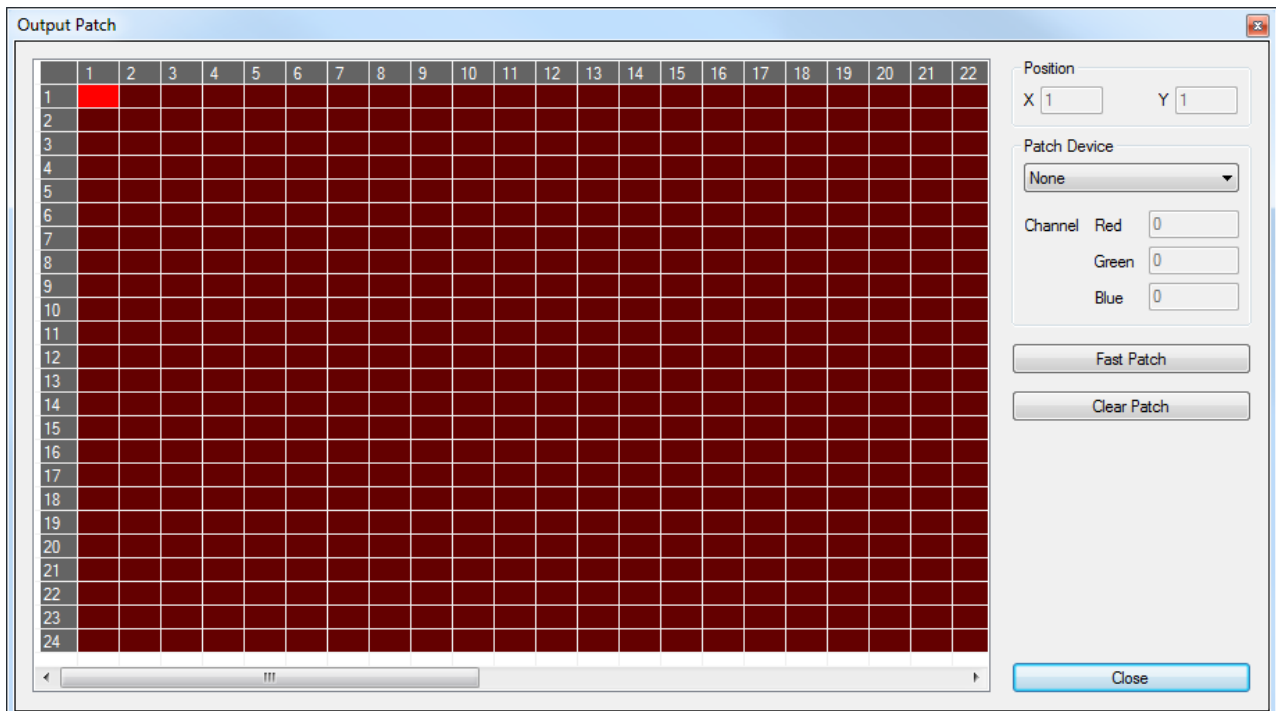
**Serial Port / USB Device:** (Empty dropdown)  
Baud: 115200

**Output Redirection:**  
 Redirect Output to File  
Select

Buttons: Cancel, OK

## Patch Matrix

When you successfully defined your devices, you should patch the matrix pixels to your corresponding output devices. This will be done in the **Setup -> Output Patch** dialog.



The 'Output Patch' dialog box displays a 24x22 grid for patching matrix pixels. The grid is currently dark red, with the top-left cell (1,1) highlighted in red. The right side of the dialog contains controls for Position, Patch Device, and Channel settings.

**Position:** X 1, Y 1

**Patch Device:** None

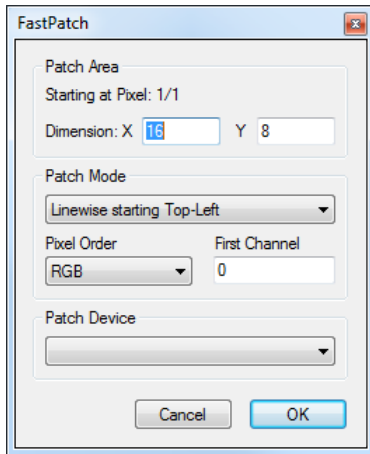
**Channel:** Red 0, Green 0, Blue 0

Buttons: Fast Patch, Clear Patch, Close

## Jinx! – LED Matrix Control

In this window you can see every single pixel of your matrix. Unpatched pixels are indicated red, patched pixels will appear in green. You can click on a single pixel and patch it to the corresponding device and channels on the right side.

To complete this job quickly, you can use the **Fast Patch** dialog.



The Fast Patch starts with the actually marked pixel. Now you have to set the dimension of the block you want to patch, choose the correct output device, the pixel order and the first device channel. To do the patch you simply have to click the OK button.

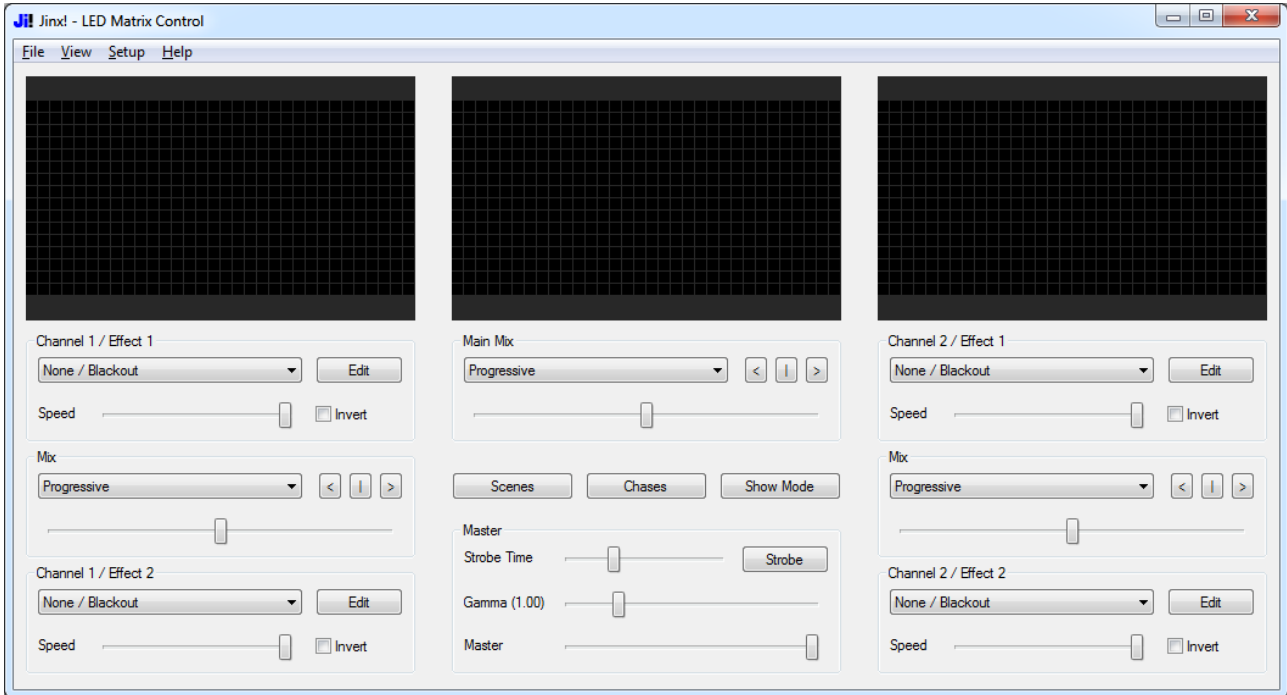
## Start Output

After patching your matrix you can start the output by activating **Setup -> Start Output**. If everything went well and you choose any effect it should be displayed on your matrix.



# Main Window

The resizable Main Window is divided into 3 sections. There are 2 effect channels, channel one is placed at the left side and channel two at the right side. In between these effect channels resides the master section, where you can control the output.

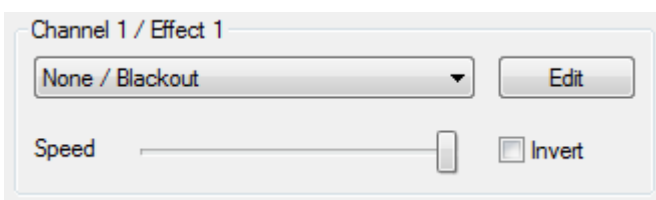


Every effect channel is divided into 2 identical effect generators and a crossfade/mix section in between where you can mix the animations that are generated by the independent generators.

The master section has another crossfade/mix section which is responsible to mix the two effect channels together. The lower control group in the master section will control the main brightness (Master), the global gamma correction (Gamma) and the master strobe, which can be used as additional effect after mixing all channels together.

## Effect Generators

Every single effect generator has 4 controls to choose and control the animation.



The most important control is the dropdown list on the top left side, here you can choose the effect that will be generated. On the left bottom you can change the generator speed, to fit your needs.

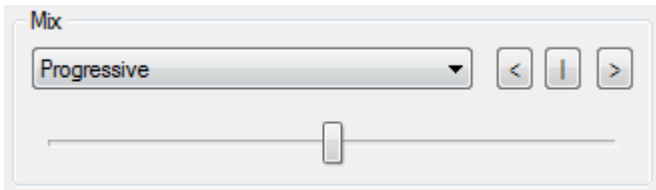
The Edit button on the right top will invoke the effect configure window, where you can change a lot of parameters for your chosen effect. The small checkbox underneath will invert the colors of the whole generated picture/animation.

## Copy and Paste Effects

You can copy an already configured effect generator into the internal clipboard by right clicking your mouse over the Edit button. You can paste it back to any effect generator by using the right click menu over any Edit button again.

## Mixing Effects

If you activated both effect generators in one channel, the animations will be mixed together. The way the mixing or crossfading will take place can be controlled by the mix section.



On the top left you can use the dropdown list to choose the mix mode. There are several modes available:

- **Progressive**  
Will do a simple crossfade of the two effects, where every effect reaches 100% in the middle of the mix fader.
- **Linear**  
A classic cross fader where every effect will reach 100% at the opposite end of the fader.
- **Upper/Lower Shape**  
With these mix modes the shape of one animation is colored with the second animation. For example, when you have some expanding squares on the upper channel, a plasma animation on the lower effect and use Upper Shape as mode you will get the expanding squares in plasma color.
- **Upper/Lower Intensity**  
Nearly the same as Upper/Lower Shape, but instead of the shape the color intensity will be used. So for example a fading or antialiasing will not get lost when coloring with another effect.
- **Upper/Lower Overlay**  
Will simply do what the name says. This will overlay one effect over the other. Every black pixel from the overlaying effect will be treated as transparent.
- **Upper/Lower Overlay (Border)**  
Same as Upper/Lower Overlay, but a small border will be drawn on the overlaying effect. So for example you can set a text with outline border over a screen filling effect like plasma or fire.

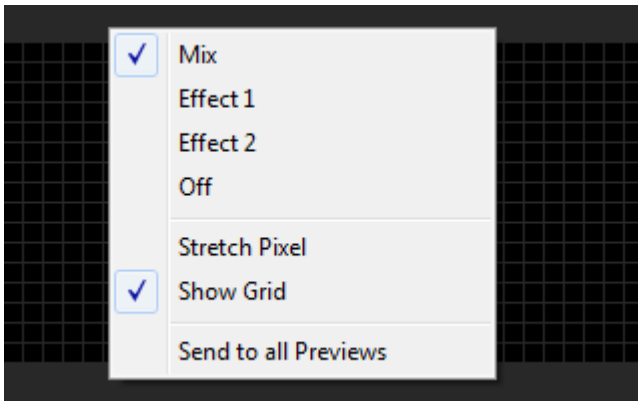
The fader on the bottom of the mix section will do the cross fade. The left position corresponds to the upper effect, the right position to the lower effect.

There are also 3 helper buttons for the fader at the top right of the section. With these buttons you can quickly move the fader to the left, middle or right position.

## Channel and Main Preview

On the top of each channel you will find a matrix preview window. Here you can see the result of your two mixed effects. To do an easy edit of one effect inside a channel you can switch this channel preview to a specific effect generator by right clicking inside the preview itself.

You can also configure the preview to show a grid or to stretch pixel. With the “Send to all Previews” option, you can set all previews (including main and showmode) to the same grid/stretch settings. The Stretch Pixel and Show Grid options are also available on the main preview.

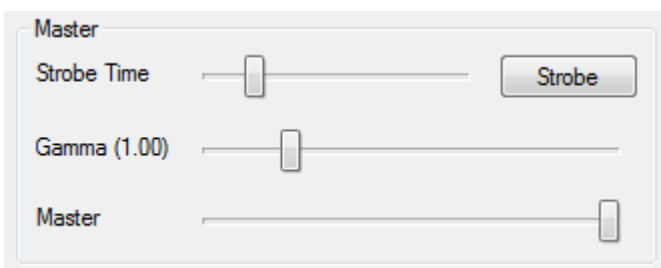


## Mixing Channels

Inside the master section you will have another mix area to mix the two channels together. This main mix will work the same way as the channel mixes, except the fader position buttons. These buttons will do a smooth auto fade to the selected position, instead of jumping directly like in the channel mix sections.

## Controlling Master Output

After mixing everything together you can take even more control over the finished animation within the master controls in the main section.



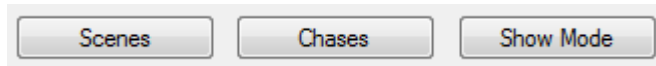
The most important fader is the Master fader at the bottom of this section. This will control the main brightness. So if your matrix is too bright into a small room you can use this fader to dim it down to an eye-friendly level. If you double click on the label, the fader will jump to 0% brightness.

If your matrix didn't have any hardware gamma correction, you can use the second fader in this section to reach matching colors on your matrix. You can set the fader back to value 1.0 if you do a double click on the label.

The top most fader in this section resides to the master strobe, which can be used as additional effect. You can set the strobe time from fast (left position) to slow (right position). To invoke the master strobe you have to push the right top Strobe button. The button will stay pushed until you click it again.

## Main Window Buttons

In the middle of the main window there are three more buttons.



With the Scenes button you will open the scenes window where you can store and manage your scenes, the second button will open the chase window where you can manage your chases. To switch Jinx! into the show mode, you have to press the third button.

More information about scenes, chases and the show mode will be found within the next chapters.

## Working with Scenes

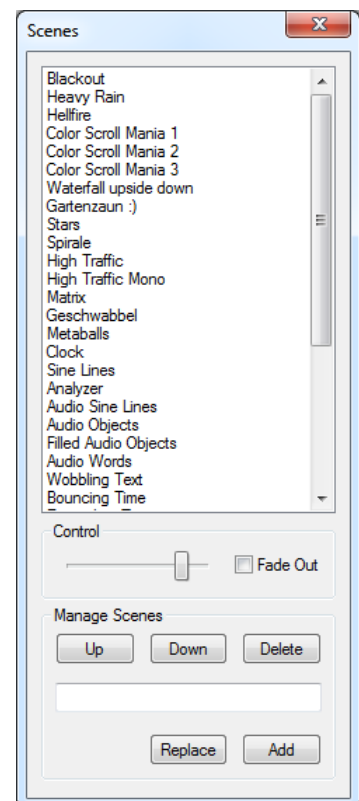
Because it is not very comfortable to design every animation mix manually when you need it, it is possible to store a complete animation setup into a scene.

The following things get stored inside a scene:

- all 4 effect generators with all effect specific options
- the 2 effect mixes
- the channel main mix

The master controls (Master, Gamma, Master Strobe) will not be saved and will operate independently from any scene. So you can adjust the main brightness and gamma to your setup or room and don't have to adjust it every time a new scene will be activated.

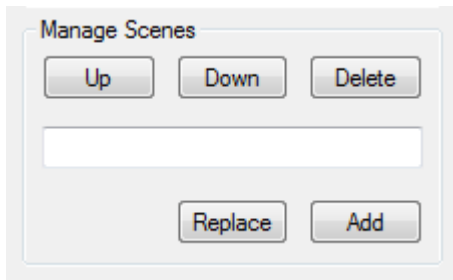
The amount of scenes which can be stored is not limited. To save, activate and manage scenes you have to go to the scene window. This can be opened over the Scenes button in the master section of the main screen or you can use the main menu. Choose **View -> Scenelist** to activate the window over the menu. Additionally you can use the **F2** shortcut to open or close the window.



The scene window is divided into 3 sections. The first and top most section is the scene list, it will display all stored scenes. The second area is called Control and will let you enable and adjust the scene fade when you activate another stored scene. The Manage Scene section will let you add, replace/rename, move and delete your scenes.

## Manage and Playing Scenes

When you successfully created your first animation you can store it into your scene memory with two easy steps.



First of all you have to give the scene a name. Just write it into the text field which will be found in between the buttons in the Manage Scenes section. To store into the scene list you simply have to click the Add button afterwards and the scene will immediately appear inside the scene list. Now you can setup another animation and store it with another name. To recall the stored scene you simply have to click on the list entry in the first section of the scene window.

**Tip:** Create always a Blackout scene with no active effect generators, so you can switch off any animation or matrix output quickly by recalling this scene.

If you want to edit a scene and change some parameters you can restore the scene by clicking it in the list, change your parameters and click the Replace button. The active scene will be overwritten with the new settings. Furthermore you can use the Replace button to rename a scene. After activating the scene you can type in a new name in the text field and scene will be overwritten with the active settings and the new name by clicking the replace button.

To copy scenes you have to activate it and use the add button after a new name was typed into the text field.

The Up, Down and Delete buttons will exactly do what you expect. With Up/Down the activated scene will move inside the scene list and Delete will remove the scene from the store. Be carefully, the scene will be deleted without any confirmation.

## Scene Fade

Inside the Control section you can activate and adjust the scene fading on changing scenes. If you activate the Fade Out checkbox the last picture from the previously running scene will be faded out while a new selected scene is already running. So you can create smooth scene changes instead of hardly switched effects. The duration of the fade out can be adjusted with the fader control beneath the checkbox from fast (left position) to slow (right position).

**Tip:** The scene window can be docked to the main window, you can activate it inside the main menu with **View -> Dock Scenewindow**. It will be undocked automatically if you move the scene window directly. You can also re-dock the window over the system menu you will get by right clicking the title bar of the scene window.

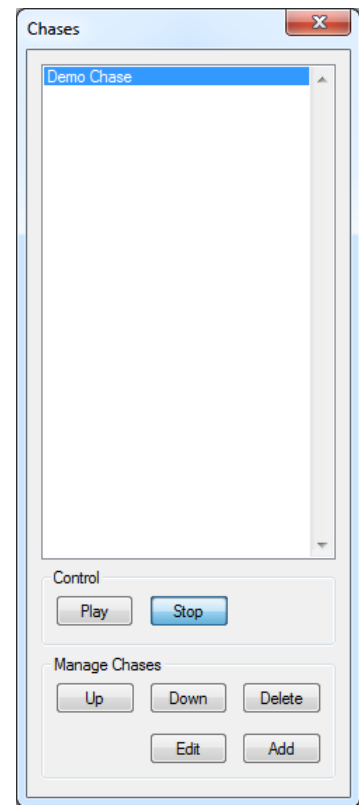
## Working with Chases

In addition to the scene store, Jinx! has also a powerful chase engine integrated to get even more flexibility.

A chase is a well-known feature within common lightning desks and it is able to playback scenes automatically. This means you can program complete shows and generate extended animations with a few easy steps.

A chase in Jinx! can use the following features:

- scene changes with adjustable scene fade for every step with fixed or randomized scenes
- control main mix cross fader with smooth auto fade or a direct jump to the wanted position
- control master strobe on/off and speed
- adjustable duration/time for every step
- unlimited amount of steps inside a chase
- the chase can be set to an endless loop



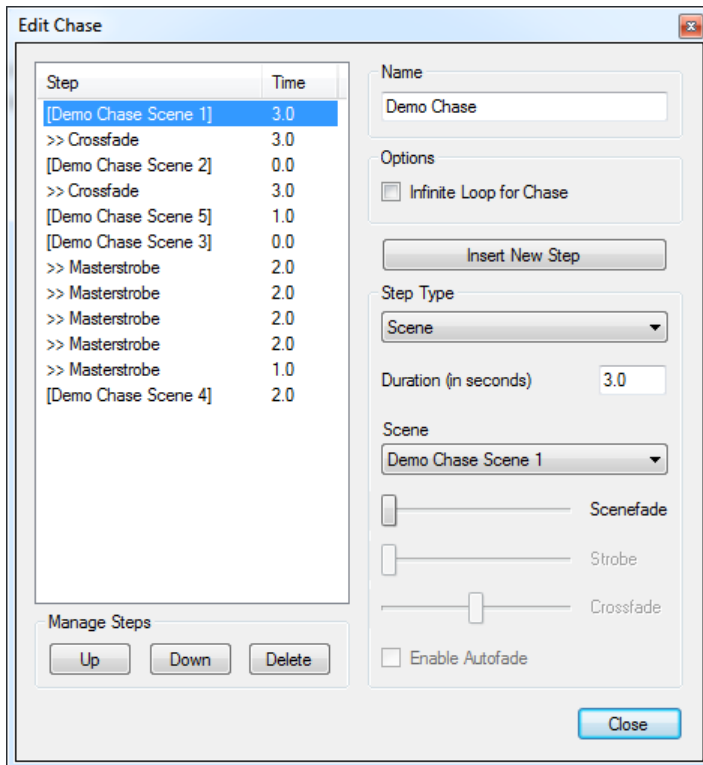
The amount of storable chases is not limited as well. The chase window can be opened over the Chases button in the main window or by choosing

**View -> Chaselist** in the main menu. You can also use the **F3** shortcut to open or close the window.

The window looks roughly like the scene window. We have the chase list at the top, a control section to start and stop chases as well as the Manage Chases section to add, edit, move and delete a single chase.

## Creating a new Chase

To create a new chase you have to click the Add button inside the chase window. A new empty chase will be created and it will be opened automatically within the chase editor.



The chase editor is clearly structured. On the left side you see the step list with the well-known Up, Down and Delete buttons below it. On the right top you can change the name of the chase and set the chase to an infinite loop, which means the chase will never stop and automatically start the first step after reaching the last.

The right section underneath the name and loop control shows the actual step and can be used to create or edit any step. To get your first step into the chase you have to click the Insert New Step button. A new step will appear on the left side and the step edit section will display the default settings for the new step. First of all you can set the step type. There are three different types possible:

- **Scene**

This will start a new scene and fade out the last picture with value you can adjust with the Scenefade fader. To switch of fading out you have to move the fader to the left position, there is no checkbox to switch off like inside the scene window.

- **Random Scene**

Nearly the same as Scene, but the scene itself will be chosen by a random generator. So you can quickly make a chase with only one step that will randomly loop thru all your stored scenes for example.

- **Masterstrobe**

The Masterstrobe event will control the master strobe. You only have one fader to do this, if you move the fader to the most left position the strobe will be switched off. The strobe speed goes from fast (left position) to slow (right position).

**Remember:** *when you switch on the strobe it will reside active until you switch it off with another Masterstrobe event. Other events in between will not change the strobe state.*

- **Crossfade**

The Crossfade event can control your main mix cross fader within a scene. You can set the cross fader directly to a position, this position will be set without any delay when this event takes place. You also can switch on auto fade with the Enable Autofade checkbox. If auto fade is active the fader only knows 3 positions: left, middle, right. When such an event starts the cross fader will fade smooth from the actual position to the given position inside the event.

**Remember:** *A new scene will set the cross fader to the value which is stored inside the scene, so the Crossfade event only makes sense after setting a Scene event.*

After setting the type and the parameters for the chosen event type you have to set the duration for this step. This is done in seconds, but you can use a decimal operator to get smaller steps (0.1). It doesn't matter if you use decimal point or a decimal comma.

**Remember:** *The time you adjust is the duration of the step, or in other words the time when the next step will take place.*

All changed values take directly place in Jinx! and you can see a preview of the step on the main window. To edit an already saved step you can simply click on it in the step list. The controls which are not useable by the chosen step type will get greyed out.

You can use the Manage Steps section to resort or delete steps. When you are finished with editing your chase you can leave the window by pressing the close button. All steps and changes will be saved automatically.

**Tip:** *You can also set the duration time to a zero value. This means the next step will start immediately. For example you can add a scene step with a zero duration followed by a master strobe event and you will get the scene starting with a strobe effect.*

**Remember:** *To avoid extreme loops, a chase will not start when the complete runtime is less than 2 seconds.*

**Attention:** *Scenes which are in use by a chase cannot be deleted anymore, you first have to delete the scene inside the chase or delete the corresponding chase. Resorting scenes inside the scene list will not break any chase.*



## Starting and Stopping a Chase

To start the playback of a stored chase press the Play button inside the chase list after selecting the wanted chase. The play button will get locked and the chase will run. To stop the chase you can use the Stop button or just start any other chase or scene.

**Tip:** You can also start a chase by using a double click inside the chase list.

**Remember:** A chase will automatically be stopped if you start or manage another chase or scene or another edit/configure action will take place (e.g. pressing effect edit, entering setup menus).

## Manage Chases

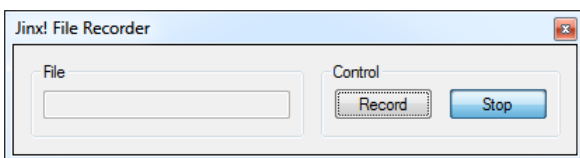
Within the Manage Chases section you can also move, edit or delete the already stored chases. You can also edit a chase with a simple right click inside the chase list.

**Tip:** The chase window can be docked to the main window over the menu **View -> Dock Chasewindow** or the system context menu by a right click on the title bar.

## The Jinx! File Recorder

Within Jinx! you can also record your created animations into single files. This is mainly used to get more possible effect combinations. For example you can record a full animation which uses all 4 effect generators inside a file. After recording it, you can use this file with the Jinx! File Player engine inside one effect generator and can combine it now with other effect generators. In that way you can combine an endless number of effects.

To open the Jinx! File Recorder you have to go to the main menu and choose **View -> File Recorder**. You can also use the **F4** shortcut to open or close the window, the file recorder can even be started inside the show mode by pressing F4.



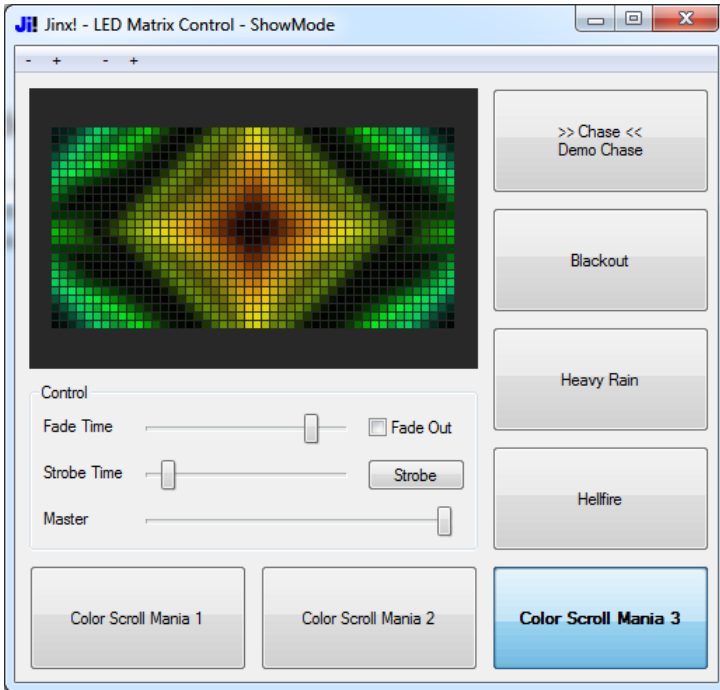
To start a recording you have to click on the Record button and a File Save window will open and ask you for the filename you want to record into. After hitting the save button the recording will start immediately. The Record button will lock and stay pressed as long as the recording is active. To end the recording, simply click on the stop button.

The Jinx! File Recorder records the main output in the background. This means that everything you will change while a recording is active will get recorded, for example scene changes or a chase playback. Even adjusting effects or adjusting the master brightness will affect the recording.

You can playback the recorded .jnr files with the Jinx! File Player engine. You can start a player engine on every effect channel. You will find more about the player inside “The Jinx! Effect Engines” chapter.

## The Show Mode

After storing and programming all your scenes and chases you can use the show mode to operate Jinx! with an easy to use interface without all unneeded controls for playback scenes and chases. The show mode can be started with the Show Mode button inside the main window, over the main menu by choosing **View -> Show Mode** or by pressing the **F11** shortcut.



With the show mode window you have a touch screen friendly screen to control all the stored scenes and chases which will be shown as buttons. The show mode window is fully resizable and the buttons get automatically placed and ordered, starting with all chases and followed by the scenes. The size of the buttons can also be adjusted with the small – and + buttons inside the top menu. The first pair will control the width and the second pair the height of all buttons.

Additionally there is a master preview window, the already known master controls for main brightness and master strobe, as well as the scene fade control. You can start any chase or scene by simply clicking the corresponding button. The button will reside pressed until another scene or chase is chosen.

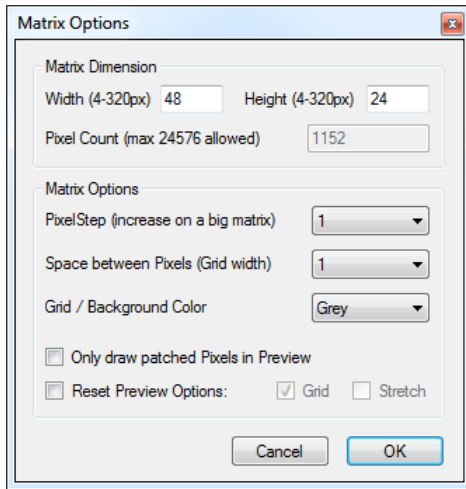
The master preview can be configured by the context menu you get by a right click on the preview itself. You can set the pixel stretch and grid as well as clone these settings to the other previews.

Close the show mode with the windows system menu (icon), the close window button inside the title bar or the **F11** shortcut and you will get back to the normal mode.

**Tip:** The button size, window position and window size will be saved when leaving the show mode and recalled the next time you start the show mode. Switching between show and normal mode will not interrupt any animation, scene or chase.

## Setup Matrix Size and Options

The first step to configure Jinx! for your matrix would be to set the resolution your matrix has. This would be done in the Matrix Options dialog. You can access it over the main menu with **Setup -> Matrix Options**.



In the first section you can set the width and height of your matrix in pixel. At the moment Jinx! supports any matrix dimension between 4 pixels and 320 pixels, with a total pixel count of 24576. This means you can choose any width and height between 4 and 320 pixel, but width x height cannot exceed 24576 pixel. For Example, if your matrix width is 320 pixels your matrix height cannot exceed 76 pixels ( $320 \times 76 = 24320$ ). You can use any resolution within the 24576 pixel limit. The actual pixel count will be displayed for your information, if your given matrix resolution is too large, Jinx! will correct it automatically.

In the lower section there are additional options to set. Because many of the effect generators are on a pixel based speed, they can get very slow if you drive big matrixes with Jinx!. To speed up those effects, you can raise the pixel step up to 4 pixel changes per frame.

The next options adjusting the matrix previews in the user interface. You can choose the width of the grid you can display in the previews (switching on off via the context menu on the previews) as well as the color of the grid and background. With the last option you can reset those preview options for all previews (including main and showmode) in one step.

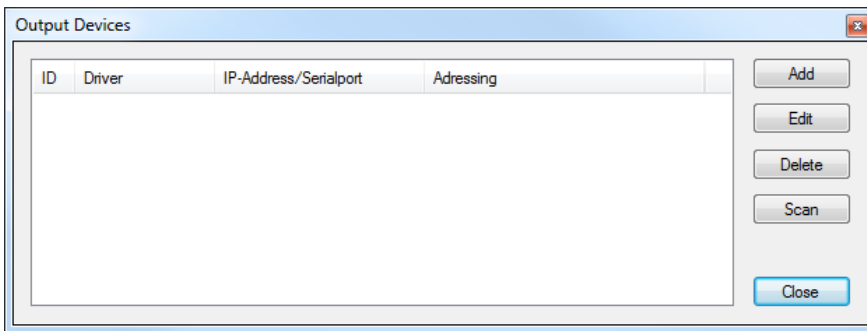
When your matrix has a custom shape instead of a normal square you can set your preview windows to the same shape of your matrix patch with the “Only Draw patched Pixels in Preview” option. So you can create custom shapes by only patching the needed pixels inside the patch window.

If your matrix is too big to display a grid within the small preview window, Jinx! will disable the grid on its own.

## Configure Output Devices

Jinx! supports various output protocols to drive your matrix. You can use multiple output devices and split your matrix over more than one interface, for example multiple Art-Net nodes or even multiple serial devices to keep the frame rate up on big matrices which uses daisy chain.

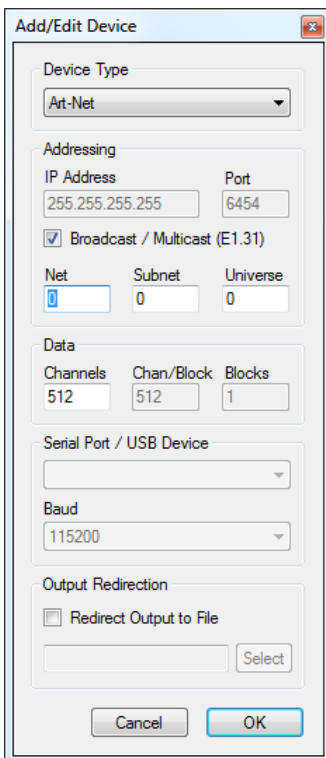
To configure your ports and protocols you have to use the main menu with **Setup -> Output Devices** to open the Output Devices window.



On the left side you can see already configured devices with its main parameters, on the right side you get the control buttons to add, edit or delete a new device. Additionally you can use the Scan button to search within your network for Art-Net nodes. Any found and useable Art-Net node will automatically be configured and added to the list.

### Add and Edit Output Devices

To add a new device simply click the add button and you will get the Add/Edit Device dialog.



This dialog has various options which are not all useable by every device type. In the first section you can choose the device type you want to configure, the next section will help you to address the device if it's a network based protocol.

The Data section will give you control over protocol specific options and includes the number of channels as well as the block size for some protocols. The next section will configure the serial port and the baud rate if it is a serial or usb (e.g. FTDI) based device and the last section will let you redirect the protocol output into a file. This would be useful to create animation files, which you can play on a standalone controller. After selecting the Redirect Output to File option you have to choose the file to record into with the select button. Recording starts on activating output and stops on closing output.

Firstly you have to choose the correct type of your device to see and set the needed parameters. The following protocols are supported by Jinx!

- **Art-Net**

Art-Net is a network based protocol, so first of all you have to set the IP address of the Art-Net node, the udp port is fixed and displayed as read only. Art-Net also needs a direct addressing inside the Art-Net network. Jinx! supports Art-Net 3, which means Net, Subnet and Universe as address parameters. If your node only supports Art-Net 2 you should use 0 as the Net address and only give the correct Subnet and Universe values.

You also can set the amount of channels to be transmitted. Art-Net, as DMX based protocol, supports maximal 512 channels per node. If you set the Channels value greater than 512 it will be corrected to 512 on saving. The Art-Net specification allows sending less than 512 channels, so you can choose lower values. The channel count must be dividable by 2.

***Remember:** If your Art-Net node supports ArtPoll, you can simply use the Art-Net scan in the Output Device dialog to find and add the nodes automatically instead of adding and configuring manually.*

- **sACN / E1.31 (streaming ACN)**

E1.31 or streaming ACN is a network based protocol to transmit DMX data like Art-Net. The protocol has been drafted and specified by the ESTA to get the new standard, replacing protocols like Art-Net. First of all you have to set the IP address of your receiver. Because the protocol was also designed for broadcasting, there is an additional address setting inside the protocol called universe. You should set the universe to fit the settings on your receiver, allowed are the universes 1-63999. Like with Art-Net, you can also specify the amount of channels you want to transmit, because it is based on DMX data, you have a limit of 512 channels. So the choose able range is 1-512 channels.

- **tpm2.net**

tpm2.net is a network based protocol designed and specified by the ledstyles.de community. It's a simple and clear structured protocol with variable frame size and based on tpm2 which will get used on serial lines.

As its network based, you have to set the receivers IP address, the udp port is fixed and displayed read only. As tpm2.net support a variable frame size you can set the channel size freely to the size your device needs. Additionally a tpm2.net frame can be split into multiple blocks, so if your device needs a special block size you should set the Chan/Block value. The amount of blocks will be calculated automatically.

- **tpm2**

The serial based version of the tpm2.net protocol with a smaller header size. As it's a serial based protocol you only have to set the frame size and the correct serial port and baud rate.

- **Glediator**

The Glediator protocol was invented and introduced by Solderlab (<http://www.solderlab.de>). It's a very simple serial based protocol which will be used by the Solderlab Matrix Controller Board. Some other devices use this protocol as well.

You have to set the amount of channels you want to transmit over the serial line. You also have to choose the corresponding serial port and set the baud rate to fit the settings of your device.

- **MiniDMX**

The serial based MiniDMX protocol is widely used and supported by many devices. You will have to set the channel count to 96, 256, 512, 768, 1536 or 3072. If you choose any other channel count, it will get corrected automatically. As its serial based, you will also have to set the serial port and the needed baud rate.

- **Enttec Open DMX USB (and compatible)**

You can use Jinx! to produce real DMX data with the help of this interface. The Enttec Open DMX USB driver inside Jinx! uses the FTDI D2XX Interface, so be sure to install the D2XX or the CDM driver. The only option you can set is the interface of your OpenDMX USB, you will get all FTDI D2XX interfaces listed which are found on your computer. The Enttec Open DMX USB will always receive a fixed channel size of 512 (full DMX universe).

- **Enttec DMX USB Pro (and compatible)**

Like the Enttec Open DMX, the DMX USB Pro is a USB to DMX Interface. For this output driver the FTDI VCP (Virtual Com Port) driver is needed, be sure to get it installed (CDM driver will include the VCP as well). Many DMX USB Pro compatible interfaces only uses Virtual Com Ports (e.g. EUROLITE USB\_DM512-PRO), so that's why we didn't use the D2XX driver model.

As with the Enttec Open DMX USB you simply have to choose the right interface port (serial port). The channel count will always be set to 512 (full DMX universe).

- **Bitmap Export**

With the Bitmap Export driver you can save the animations as a series of bitmap images. This driver can be used additionally to any other patched output and doesn't need to be patched. If this driver is active you have to choose the destination directory with the select button at the bottom of the dialog. After starting the Output every frame (25 frames per second) will be saved to this directory as Windows Bitmap file with an ascending number as filename. Be sure to have enough disk space available.

You can use external software like bmp2avi or something similar to create video/animation files out of the bmp series.

- **Glediator File Recorder**

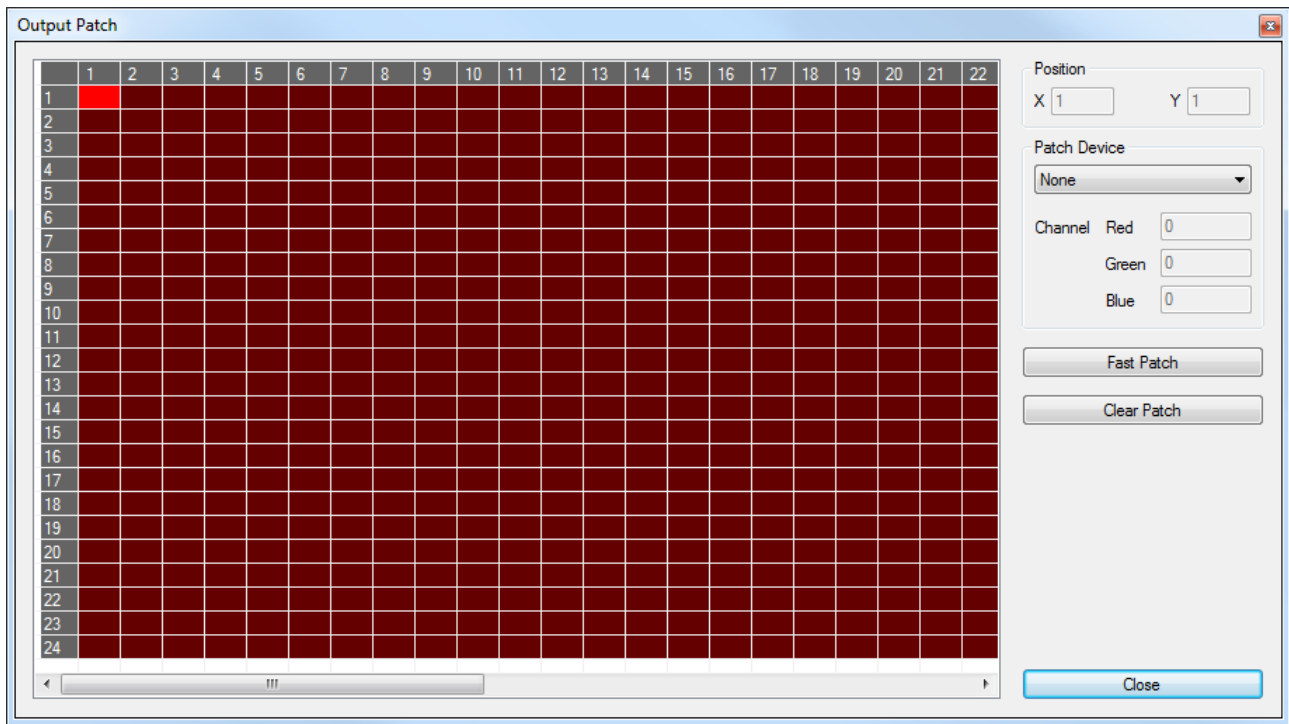
This driver can produce Glediator2 / Solderlab UIB compatible recording files. These files can be played inside Glediator2 or used within the upcoming Solderlab UIB Board. After choosing the out file you can start/stop the recording by activating/deactivating the output. There is no patching needed for this device type and so you can use it, like the Bitmap Export, additionally to any other active output patch.

**Tip:** Jinx! support broadcasting for the Art-Net and multicasting for the sACN/E1.31 protocol. So if you have a small network and your Art-Net or sACN/E1.31 nodes are configured well, you can set activate the Broadcast/Multicast option and don't have to take care about any IP parameters.

## Patch Matrix

When you added and configured your output devices, it is time to patch the single pixels to the corresponding devices and channels. Jinx! give you control over every single pixel and you can patch it to any defined output device.

The patching takes place in the Output Patch window, which can be opened through the main menu over **Setup -> Output Patch**.



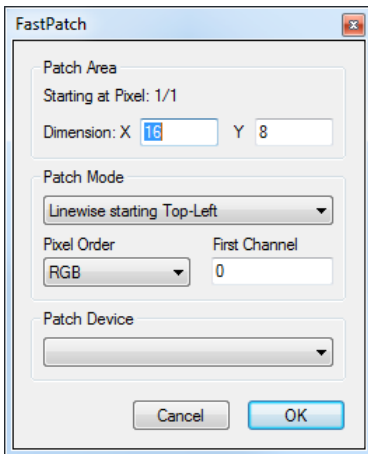
Inside this window you can see and scroll through every single pixel of your defined matrix. Every pixel will show you its actual state. If it's marked red its unpatched, means no output device and channel is assigned, if it's already patched it will be marked green.

You can see and edit the actual pixel assignment on the right side of the window. If you click on any pixel you will see the position in coordinates as well as the assigned output device and channels for the three colors red, green and blue.

After activating a pixel, you can edit the device and channel assignment and it will be stored immediately. To get this work done more comfortably, than editing every single pixel, you can invoke the Fast Patch window.

## Fast Patch

The fast patch dialog will help you to patch your matrix in a very short time. To open the dialog, use the Fast Patch button on the right side.



The fast patch will assign ascending channel numbers within a device to the next pixel, corresponding to your matrix controller and setup you can patch the whole matrix at once or single areas.

On the top of the fast patch window you will see your starting point, this will be chosen by the activated pixel in the output patch window. You also have to tell Jinx! the size of the area you want to patch. For example the complete matrix size, or if your matrix is assembled with single boards the size of a single board.

The section Patch Mode will define how your pixels are latched together, you can choose all common modes with the drop down list. You also have to set your color order inside a single pixel and the starting channel on the corresponding output device as well as the output device itself.

As soon as you press the OK button, Jinx! will enumerate your pixels to the output device and channels in the selected order and will return you to the output patch window.

If everything is done you can use the Close button to leave the output patch window.

**Tip:** The fast patch area always is defined from the top left and will cover the area to the right and to the bottom with the given size, even if you choose a patch mode which will start at the bottom.

**Attention:** If you change your output devices, the patch will be corrected if necessary. So if you need to edit your output devices after patching, take a look at your patch again to be sure that everything is assigned correctly.



## Starting Output

After configuring the complete output section, you will have to start the output devices to get the animations transferred to your matrix. You can simply start the output with the main menu entry **Setup -> Start Output**.

**Attention:** Jinx! will always produce 25fps. So be sure that your output device is able to handle it (e.g. serial line with a too small baud rate), otherwise a frame skip will appear.

## Remote Control

Jinx! can be controlled remotely by various protocols, so you can seamlessly integrate it into any lightning installation. The remote control feature can be enabled and configured over the main menu with **Setup -> Remote Control**.

The remote control supports four different controls and will receive and interpret 8 bit values on every channel. The following controls are available:

- **Scene Select**

Jinx! will map every received number to the corresponding scene in the scene list. The value 0 will be ignored. For example, if the received value is 1 the first scene would be activated. You can choose any scene between 1 and 255.

Additionally you can limit the scenes to 32. In that case the values 1-7 will activate the first scene, 8-15 the second scene and so on. The value 0 gets ignored in that mode as well.

- **Chase Select**

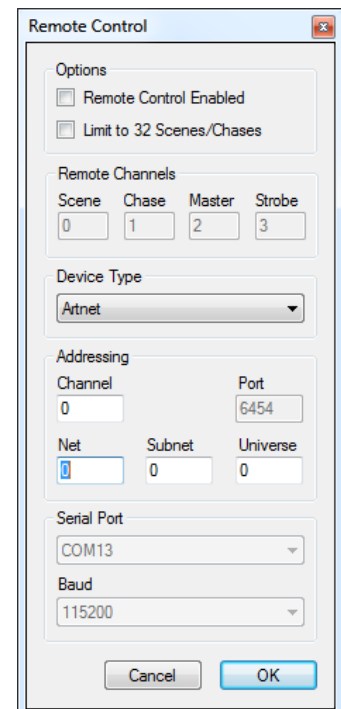
The chase select will operate the same way as the scene select. You can choose and start any chase between 1 and 255. The 0 values will get ignored as well. If you activated the scene limit to 32 scenes, the chase selection will also get limited to 32 chases.

- **Master Brightness**

With the master brightness control you can set the main brightness. Accepted values are from 0 to 255, where 0 means 0% and 255 will get out 100% brightness of your matrix.

- **Master Strobe**

This control also takes values between 0 and 255. The value 0 switches the master strobe off, 1-255 will activate the strobe. Smaller values will give you a faster strobe.



In the first section of the remote control dialog you can enable or disable the remote control feature by setting the first checkbox. You can also activate the scene / chase limit in this dialog section.

## Jinx! – LED Matrix Control

The second area will display the 4 controls and tell you the actual channel they are using. The channels will always be ascending and the control order is fixed. The start channel can be set inside the protocol addressing area.

**Attention:** *Jinx! will always start any channel count at zero. DMX normally starts counting in human order, means 1 is the first channel. So if you converting any DMX data to remotely control Jinx! you should take care about that and know that DMX channel 1 will get Channel 0 inside Jinx!.*

You can configure the wanted remote control protocol with the device type drop down list. The following protocols are available:

- **Art-Net**

The network based Art-Net is also accepted to get remote commands. You will need to assign the Art-Net addressing with Net, Subnet and Universe. The start channel must be assigned as well. Jinx! interpret every addressed Art-Net frame, with sizes between 4 and 512 channels. The Art-Net implementation also supports ArtPoll, this means any lightning desk or software which is able to do ArtPoll can find and configure the Jinx! remote automatically.

- **sACN / E 1.31 (streaming ACN)**

You can also use the network based sACN/E1.31 protocol to receive remote control data. You have to the universe Jinx! should listen to (1-63999). Jinx! will receive and use every sACN frame with a minimum of 4 channels and use the given start channel.

- **Enttec DMX USB Pro (and compatible)**

The Enttec DMX USB Pro interface supports DMX IN to your computer (not all compatible do this as well), so you can use any hardware based DMX Desk to remotely control Jinx!. You have to configure the correct serial (Virtual Com Port) of your device and set the start channel Jinx! should look at.

- **tpm2**

For the serial based tpm2 protocol (see Configure Output Devices for details) you need to set the incoming serial port and the corresponding baud rate as well as the start channel. Jinx! will receive every tpm2 frame with a minimum of 4 and maximum of 512 channels and will use the given start channel for the first control.

- **tpm2.net**

As tpm2.net is a network based protocol you only need to set the starting channel inside the received frames to map the control functions. Only frames between 4 and 512 channels will be accepted.

**Tip:** *To control Jinx! with your DMX lightning desk, you can also use any DMX2Art-Net node or use a small SEDU based solution, which will do a DMX2tpm conversion (<http://www.sedu-board.de>).*

## Audio based Effects

Jinx! also offers some audio based effect generators, which will be driven by a Fourier analysis to get frequency controlled triggers. The audio capturing will be done automatically and always uses your default windows capture/recording device. The selection and configuration of this device depends on your windows version and your audio hardware. You will find information inside your audio device manual.

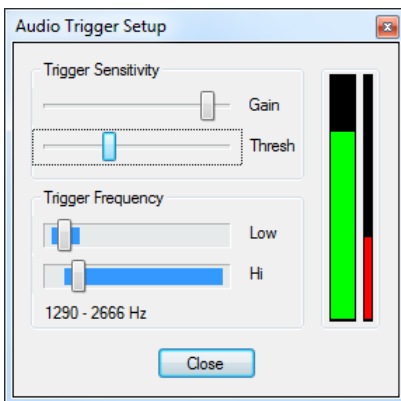
**Tip:** If you playback your music with the same computer that runs Jinx! you should configure your audio device to capture the main output. If your soundcard doesn't offer this feature, you can use the free available VB-Audio Cable driver.

### Auto Gain Control

The audio engine inside Jinx! has a built-in auto gain control. This means the engine will try to normalize the captured audio to a common level. If you activate this feature you are able to get the same effect results, even if your audio volume changes. You can enable and disable the auto gain control within the main menu over **Setup -> Audio AutoGainControl**.

### Audio Trigger Setup

The multi-frequency based effects like spectrum analyzer get configured automatically and the only audio control available is a gain fader. For other effects, for example strobe or expanding objects, an audio trigger has to be configured. The trigger setup will be invoked when you press the corresponding trigger setup button inside the effect configuration.



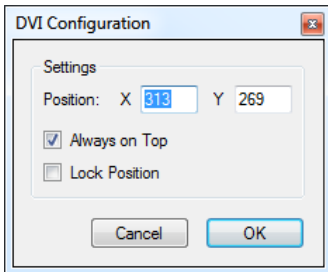
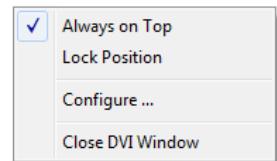
Every audio trigger can be configured to a frequency range within it will analyze the audio data. You can adjust the low and hi shelf frequencies with the 2 faders at the bottom. On the right side you see two level bars. The first level bar will show you the actual audio level within the given frequencies. You can influence this level with the gain fader to get a good level. The second smaller bar shows the actual threshold of your trigger. It is adjustable with the thresh fader. Adjust your threshold to get the audio based effect work. The threshold bar will change its color from green to red when it gets triggered, which means the audio level is above your chosen threshold.

## The DVI Output Window

Jinx! also provides a borderless DVI Output window, which can be placed on screen to get the matrix data out over your dvi port to dvi driven matrix controllers. The window is place able on all connected screens and can stay always on top. It will also wipes out any mouse cursor/pointer to get the graphics output onto your matrix not distorted.

You can activate this window over the menu with **View -> DVI Window** or via the shortcut **F9**. The window will open with your matrix resolution and will be placed at the upper left corner of your screen. You can move it to any other location by left clicking and moving it. The Window position will get saved and restored after you load your show again.

You can further configure the window with the context menu you will get with a right click inside the window. You can set the Always on Top flag as well as lock the position of the window. This means it is not moveable unless you deactivate the Lock Position feature. Further you can start the DVI Configure window.



The DVI Configuration window can also be started over the menu with **Setup -> Configure DVI Window**. Here you can set the Always on Top and the Lock Position features as well. Additionally you can set the position of your DVI Window exactly by given the corresponding pixels. The pixel count will start with 0 on the upper left corner.

## Saving and Loading your Show

Jinx! will manage and save all your settings, scenes and chases in one single file, so you can load your show with one single step. The following things will be stored inside the .jnx files:

- complete setup with Matrix size, output devices, patch, remote control
- all single menu options like auto gain control, windows docking
- all main section settings like strobe time, master brightness and gamma
- show mode window size and position as well as button sizes
- all stored scenes including actual scene fade setting
- all chases stored in the chase list
- input/output states, if the output or remote control is enabled while saving it will be automatically started, if possible, after loading that file again

You can simply load and save files over the main menu entries **File -> Open** and **File -> Save**.

### Importing a Show

If you already configured or loaded a complete setup and will only import scenes and chases from another show, you can use the **File -> Import** menu entry. Jinx! will only import scenes and chases with this function and add it to the actual scene and chase list, no other settings will be loaded and no already stored scenes or chases in the lists will get overwritten.

***Tip:** Save a complete setup with some basic scenes for every matrix or matrix setup you have. So you can do a quick start to match your hardware and are able to import additional scenes and chases from a global file.*

### Auto save and load

There is also an auto load feature integrated. Whenever you exit Jinx! it will save the actual state to an autosave.dat file to your Jinx! directory. The next time you start the software, this file will automatically be loaded and you can start with your last state. You can delete the autosave.dat at any time to get a clean startup.

## Command Line Options

There are several options to auto start a scene or a chase at startup as well as invoking the show mode and loading show files.

### Loading Files over the Command line

Jinx! can load your show over the command line, so you can make different startup links to start with different shows or you can assign all .jnx files within windows to start your show file with a simple double click. To start Jinx! with a specific file, simply give the file name with path on the command line. For example: `"jinx.exe c:\shows\show.jnx"`

### Auto start a scene

You can auto start any scene within the auto load show or a given show file by putting the number (counting starts at 1) with the parameter `-s` at the command line. This will start the first scene from the given show for example: `"jinx.exe -s1 c:\shows\show.jnx"`

### Auto start a chase

Auto starting a chase will work the same way as scenes with the parameter `-c` at the command line. This will start the second chase in the file: `"jinx.exe -c2 c:\shows\show.jnx"`

### Invoke the show mode at Startup

You can also invoke the show mode on startup with the `-m` parameter. This will start the first scene from the file and start the show mode for example: `"jinx.exe -s1 -m c:\shows\show.jnx"`

### Limit output frame rate to 20fps

Additionally you can use the `-p` switch to limit the output frame rate to 20fps. This option is for debugging purposes and not designed to use in a production state. The internal generators will work with 25fps regardless off the `-p` switch.

**Attention:** When you combine `-s` and `-c` on the command line, the chase will be preferred if available.

**Attention:** When a scene or chase is given at startup and it exists, Jinx! will try to auto start the output as well, even if it has a stop state inside the show file.

## Keyboard Shortcuts

You can use the following shortcuts to access various functions.

- F1        You can press F1 in any window or dialog inside Jinx! to get the context sensitive help.
- F2        Open or close the scene list window.
- F3        Open or close the chase list window
- F4        Open or close the Jinx! File Recorder.
- F9        Open or close the borderless DVI output window
- F11       Start or Exit the show mode window.
- F12       Display output frame rate (fps) within the main or show window title
- ESC       You can use ESC to quickly close all dialogs.

## The Jinx! Effect Engines

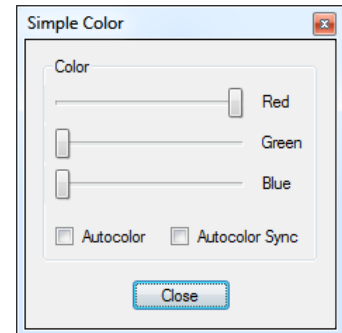
Jinx! will give you various effect engines to create your animations. Every engine comes with a lot of parameters and some effects will give you sub-effects as well. Take a look at the demo scenes to get a feeling what can be done by adjusting and combining these effects.

### Simple Color

This most basic effect will give you a simply colored matrix. It is useful for coloring other shape based effects or to coloring simple black & white gif animations.

Within the configuration dialog you can adjust and mix the color with the three base color faders: red, green and blue.

Additionally you can activate the Autocolor function which will fade the color through the whole rgb color space starting at a random value.



The Autocolor Sync option will synchronize the auto color to other effects, which has the same option enabled.

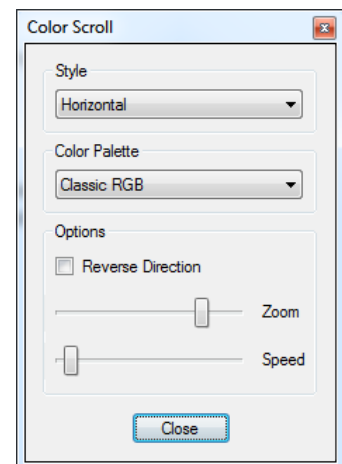
**Remember:** The color controls including auto color will be seen at many other effects and always will work the same way.

### Color Scroll

The Color Scroll engine is a very powerful effect. You can produce various color moves and additionally use it as a shape generator with the black/white color palette. You will find some examples inside the demo scenes.

You can choose different styles like horizontal, vertical and diagonal color fading as well as a shape based fading like a circle or diamond.

Additionally you can control zoom and speed of the color scroll and reverse the scroll direction.



### Plasma

The Plasma effect is a classical effect, which has been used in computer graphics for a very long time. You can adjust the effect by setting the plasma type, the color palette as well as the zoom level. The plasma engine also has an own speed slider to get a smooth fading on small matrices.

### Fire

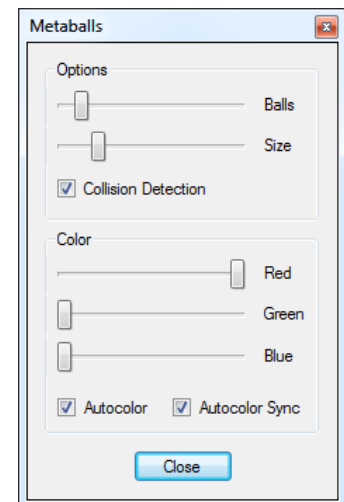
This is another basic effect which will give you a great background or standalone wall of flames. You can adjust the size and amount of the hotspots which will drive your fire.



## Metaballs

Metaballs are a well-known effect since the first days of computer graphic demos. The bouncing balls, which will merge to a wobbling thing on hitting themselves, will give you a great standalone effect.

You can control the amount and the size of the balls, as well as the color. The color section in the configure dialog is already known and has been introduced by the Simple Color effect.



## Expanding Shapes

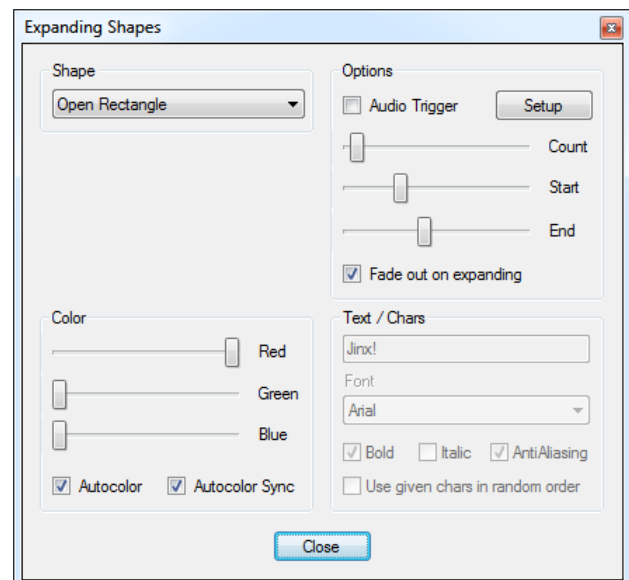
The Expanding Shapes will give you a lot of options to generate stunning effects. This engine will generate shapes at a random position and will expand the size while fading them out. You can choose various shapes like rectangles, circles, or five-stars in an open or filled variation.

Additionally you can use words or characters instead of simple shapes and also trigger this all with an audio source.

The most important control will be the Shape control, here you can choose all the shapes or switch over to the text/chars mode of this engine.

The main control will be the Count, Start and End fader. After adjusting the amount of shapes with the count fader, you can select the start size and the end size of the expanding. Additionally you can turn off the fading of the expanding shapes.

If you activated the text/chars mode, you also can control the font and the text. You can use whole words to expand or switch over to use single chars with the last option (Use given chars in random order). You will also find the well-known color section.



As already said, this effect can be controlled by audio. This means with every audio trigger a new shape will be started and begins to expand and fade. After activating the audio trigger you have to setup your trigger frequency, threshold and gain. You will find more about this in the "Audio based Effects" chapter inside this manual.

## Falling Rain

This engine will generate falling drops which will fade out to a line. You can control the amount and the length of the drops as well as the moving direction (top->bottom, bottom->top, left->right, right->left). The already known color control is also available.

## Radar/Scan Lines

Another simple line based effect which will produce a moving line, which looks like the knight rider effect or a radar scan line. You can choose the width and direction of the line as well as a fade out/tracing effect. The color control section is also available. You can additionally set an starting offset to get a defined starting point on loading scenes from the list.

## Scrolling Text

This will be one of the most used effects inside Jinx!. You can produce a scrolling text with many different styles and options.

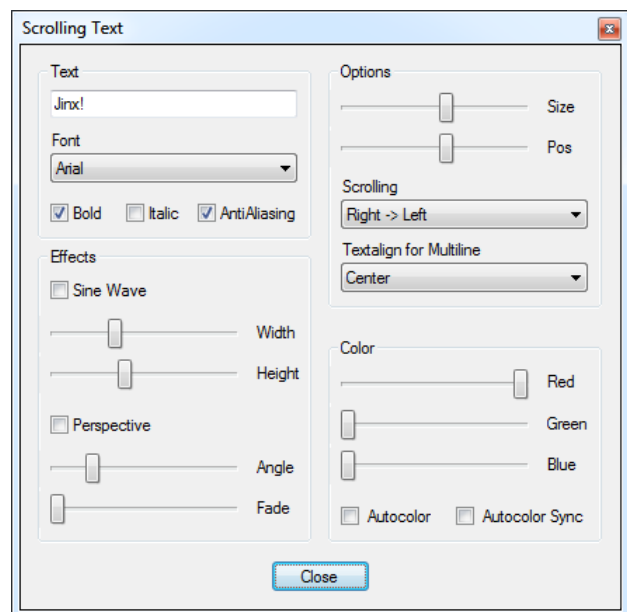
After entering your text, you can select the font and the font attributes like italic or bold. The text can also be smoothed by using the anti-aliasing feature.

With the Size and Pos fader you can adjust the text to fit into your matrix. The text scrolling can be set to different directions. You also can use vertical scrolling to get a typical movie credits effect. When you use multiple lines you can additionally set the text alignment to left, center or right.

To get multiple lines, especially for the vertical scrolling, you can use `\n` inside the text for line breaks. To get even more attention to your text, you can activate a sine wave, which will transform and bounce your text over the matrix. The well-known color section will complete this configuration screen.

Another available effect will be a perspective transform, mostly known as star wars effect. This will scroll your multiline text into a 3d space. You can adjust the angle and the fade out when using this effect.

**Tip:** You can use the scrolling text effect to display the actual time or date. When you use inside your text the placeholder `$TIME`, it will be replaced by the actual local time with hours and minutes (hh:mm). You can also use `$LTIME` to get an extended time format with seconds (hh:mm:ss). When you want to display the actual date you can use `$DAY`, `$MONTH` and `$YEAR` to build your date string (e.g. `$DAY.$MONTH.$YEAR` or `$MONTH/$DAY/$YEAR`).



## Image Viewer

To get custom effects onto your matrix, you can load and display gif pictures and animations. This engine will play animated gif files in an endless loop or simply display single frame gifs or other image files. You can resize the picture to fit onto your matrix. When your image file has another aspect ratio than your matrix, you can choose if you want the picture to be stretched over the whole matrix or if you want to keep the aspect ratio. With the aspect ratio zoom you can decide if you scale the complete image inside your matrix or if you want to fill the complete matrix and crop the parts that don't fit. You can also center the picture inside the matrix, if the displayed picture is smaller than your matrix resolution.

Additionally you can activate the bicubic resize algorithm which will give you a much smoother image resizing, but will cost you more cpu time. With the last option you are able to convert your image to a greyscale picture. This can be useful when the image will be combined with other effects.

With the Slideshow mode you are able to automatically display images from a given folder. You can trigger the image change by time, by an audio trigger or a combination of both. If audio and time triggers are enabled the image change will take place at the first audio trigger after the given time has passed by.

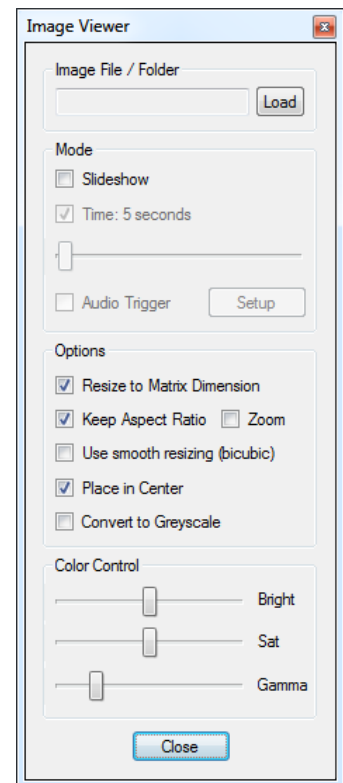
With the Color Control section you can adjust the brightness and color saturation of the image and do an additional gamma correction. This is useful to balance out some the small color and brightness loss you will get when doing a smooth resize. You can always get back to the initial values inside the color control by double clicking the labels (Bright, Sat, Gamma).

The following image file formats are supported: gif, jpeg, bmp, png, tiff

**Attention:** Be careful on using the bicubic resize algorithm, it can give you some small freezes if you use images with a really big resolution. Try to use images with a resolution that matches your matrix size instead of using images with a lot of megapixels to avoid such problems.

## Starfield

There is not much to say about this effect generator. It will produce a 3d star field effect, well-known from many computer games and movies. You can control the amount of stars and the speed you are using for your flight. Additionally the star field can be zoomed in or out. Coloring will be done with the standard color control.



## Fading Pixels

Another simple effect engine which will display single pixels at random positions, which will simply fade out. This can produce nice distorted looking backgrounds for other animations for example.

You can adjust the amount of pixels as well as the fade out speed and the size of the pixels. In addition to the well-known color control you can activate random colors for every produced pixel.

## Simple Lines

The Simple Lines module produces line based effects. There are some sub effects available with various options. The line drawing sub effects Cross and Rectangular don't need any specific options.

For the Dancing Lines and the Dancing Snake effects, the amount of lines or snakes can be chosen with the corresponding roll down list.

The Spotlight effect will be controlled by the Beam and Phase values. With the beam fader you can adjust the width of the spotlight beam and the phase fader will give you the possibility to get a phase drifting on multiple spotlights. You can additionally activate multiple spots placed in the corners of your matrix with the checkboxes inside the Spotlights section.

For all sub effects a fade control is available which will fade out the last position and gives you a motion blur like, smooth moving effect.

The standard color control is also available.

## Sine Lines

This effect will produce moving sine waves on your screen. You can choose the amount of lines (1-3) as well as the width of your sine wave. Every sine line has its own color control and can be set independently.

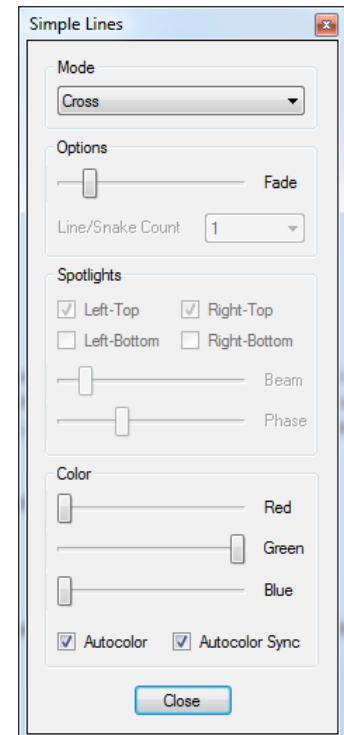
In addition to the standard mode, this effect can also be used as a 3-band audio analyzer. After activating the audio trigger checkbox, every enabled sine wave will response to a specific audio frequency. You can adjust the audio response of the sine wave amplitudes with the audio gain control.

## Strobe

The Strobe effect will flash your matrix so it can be used as an audience blinder. It is also helpful to strobe other effects with the various effect merge options. The strobe can be controlled by two time faders. The first one will adjust the time of the flash, the second one adjusts the blackout time of the strobe.

Additionally the strobe can be triggered with an audio signal. After activating the audio trigger checkbox, you have to setup the trigger. You will find more information about the audio trigger in the "Audio based Effects" chapter inside this manual.

The strobe can directly be colored with the already known color section.



## Spectrum Analyzer

This engine will display a classic spectrum analyzer on your matrix. The amount of frequency bands can be adjusted to anything between 4 and the pixel width of your matrix. The available frequency bands get limited to 64, if your matrix width is greater than 64 pixels.

The frequency for every band/bar will automatically be calculated over a logarithmic scale.

Every frequency bar is displayed in 3 colors. You can adjust these colors independently as well as their beginning with color transition pos faders. With the Color Fade option the colors will be smoothly faded between the positions.

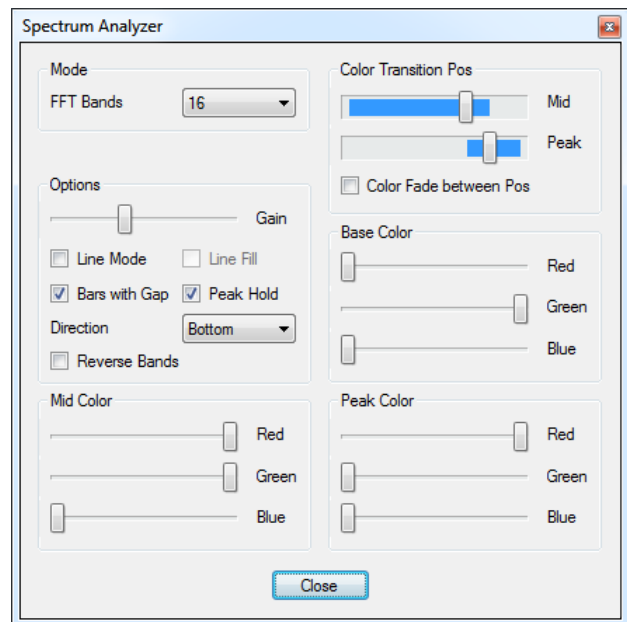
The height of the bars can be adjusted with the audio gain control.

There are multiple options available for the spectrum analyzer. You can get a clear picture by activating gaps between the bars and even can enable or disable a peak hold feature.

If you don't want Jinx! to display frequency bars you can enable the line mode. Here you will get a single horizontal line which responds to the audio bands as well. You can also fill the area underneath the line to get a more matrix filling effect. Enabling peak hold will give you a second line in this mode.

Additionally you can choose the direction the bars will follow as well as reverse the bands, which means you will swap the high and low bands to get a mirrored display.

**Tip:** Use less frequency bands than pixels when using line mode and the line will be drawn more smoothly. A good starting point will be half the amount of your matrix width.

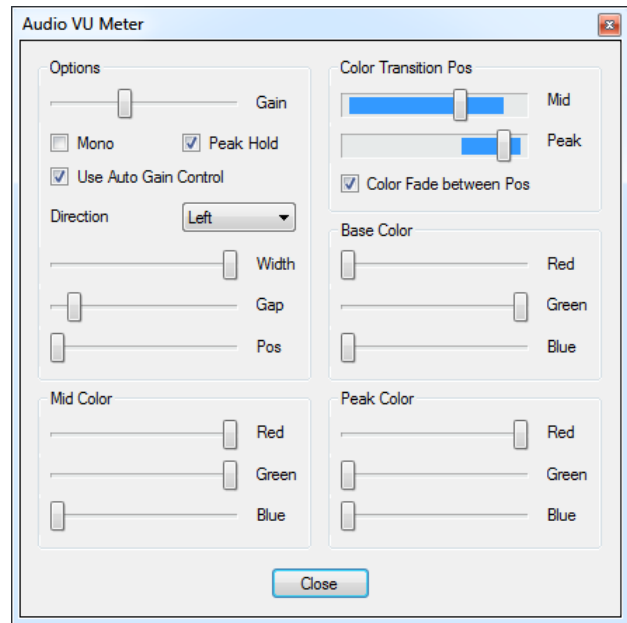


## Audio VU Meter

The Audio VU Meter is very similar to the spectrum analyzer, but it will simply display the overall rms level.

The bar design is already known from the spectrum analyzer. Every bar will use 3 colors, which you can set on your own. The color transition positions can be adjusted as well as you can activate the color fade between those positions.

Inside the main options you can activate the peak hold function and you can switch the Audio VU Meter to mono, so that only one bar will get displayed. Additionally you can turn off the audio auto gain control for this single engine. So you can display a real vu meter, while the spectrum analyzer will still produce full screen bars with corrected gain.



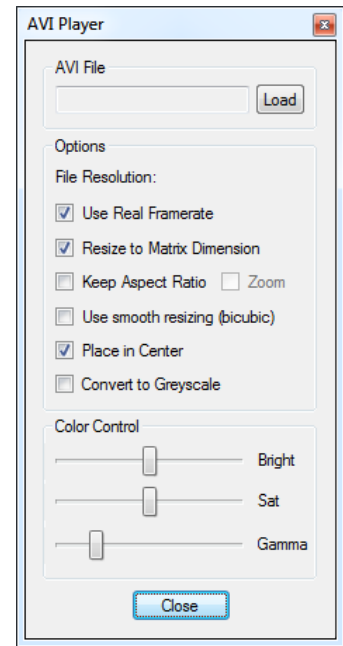
With the direction option you can choose the direction your bars will follow on your matrix. To use the VU Meter as additional effect to use as overlay, you can define the width of the bars as well as the gap between (in stereo mode) and the position on the matrix. So it is possible to create small bars which will stay at the edges of your matrix (with a huge gap) or you can use really huge bars as full screen. You can even use a single bar to animate a led stripe.

## AVI Player

The AVI player will give you the possibility to directly play avi files on your matrix. The avi player is based on the Video-For-Windows API and is able to playback all VfW-codecs that has been installed on your windows.

Inside the config window you can load the avi file and adjust a few options. You can resize the video to your matrix dimension and take control over the aspect ratio. You can also use the already known smooth resize to get better results on downsizing. Additionally you can convert the whole video to greyscale or use the Color Control section to adjust brightness and color saturation as well as doing an additional gamma correction. To get back to the initial values you can do a double click on the labels (Bright, Sat, Gamma).

With the default options, the AVI player will use the real frame rate which is given by the AVI file itself and it will also disable the speed control for this effect channel. If you want to set the speed by yourself, you have to deactivate the Use Real Framerate option and you will be able to set the speed with the effect channel speed control.



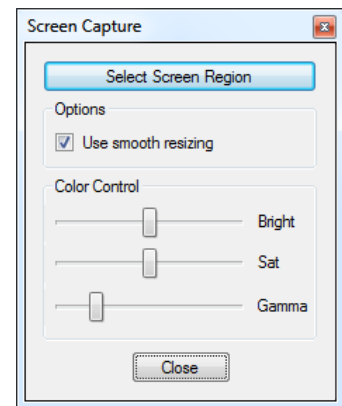
**Tip:** A good codec package will be the *ffdshow-tryout* distribution. Please be sure to activate the “VfW Interface” option while installing.

**Attention:** Decoding the video and resizing is very cpu intensive, so it is not a good idea to use something like a full hd video and size it down to small matrix.

## Capture Screen

The Screen Capture engine will give you the possibility to show everything on your matrix you want. It will capture a defined region within your screen, so you can capture and display for example a media player driven audio visualizer, flash animations or simply a movie.

The screen capture will start immediately. To define the screen area you will get a small resizable window when you invoke the “Select Screen Region” button inside the effect configure. Place and resize this window to fit exactly on the screen area you want to capture. After closing the dialog you should see the result on your matrix. The aspect ratio will be defined by your matrix resolution.



You can also choose if the resizing should be done smoothly or if a simple resize, which needs less cpu time, will be done. To compensate the color loss by a resizing you can adjust your screen capture with the brightness, color saturation and gamma controls. You can get them back to the initial values by double clicking on the labels (Bright, Sat, Gamma).

The screen capturing is limited to one effect generator, so you can only choose and active this engine once.

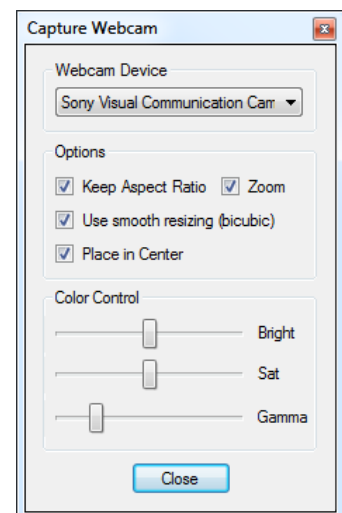
**Tip:** Screen capturing within windows can be cpu intensive. Windows has to synchronize the screen and all graphical effects within the desktop to capture the screen. So if you are working a lot with screen capturing you should disable all windows effects, especially aero, to get less cpu load and a smooth screen capturing.

## Capture Webcam

To display live videos on your matrix you can use the Capture Webcam Engine. This Engine will start up any webcam that is directly connected to your computer and will display the live video on your matrix.

The picture can be controlled by the already known resize and color options. You can active the Aspect Ratio switch / zoom as well as the bicubic resize algorithm. The brightness, color saturation and gamma correction can be adjusted with the given controls.

The Capture Webcam engine is limited to two effect generators. After activating these generator there will be no camera active, you will have to choose one inside the config screen first.



**Attention:** Starting and stopping a capturing device takes some time, so be patient. A few seconds will pass until you will get your picture displayed.

**Tip:** If you want to use an ip network camera you can use a small direct show filter which was created by Roman Ryltsov to use mjpeg capable ip cameras as direct show device. You will get more info at <http://alax.info/blog/1216>



## Jinx! File Player

With the Jinx! File Player engine you can playback .jnr files, which you can record with the Jinx! File Recorder. You will find more about it in the chapter “The Jinx! File Recorder”.

To load a .jnr file you have to click on the Load button and a file open dialog will appear. After selecting a file, the recorded matrix resolution will be displayed. If it doesn't match your actual matrix resolution the recording will be automatically resized. Additionally you can activate the smooth bicubic resize to get better results on downsizing.



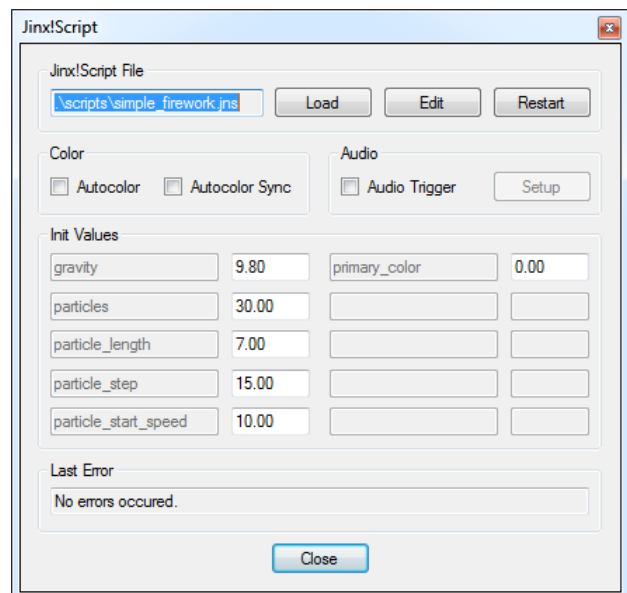
## Jinx!Script

Jinx!Script is small but powerful basic like programming language to create your own effects. The Jinx!Script engine uses a byte code compiler and a small virtual machine to run your code inside Jinx!

You will find more about Jinx!Script and a complete language reference as well as sample codes inside the chapter Jinx!Script Reference.

The engine itself can be controlled with the config screen. Here you can load the script file you want to run. Additionally you can invoke a text editor to work on the script with the Edit button. If you changed anything inside the script file you can reload and restart it immediately with the Restart button.

With Jinx!Script you have the ability to use autocolor values inside your code as well as an audio trigger. When you want to use these features inside your script, you have to activate it and configure it.



The next section will display you user configurable variables. Inside your Jinx!Script code you can define start variables as config variable and it will get displayed here. These config variables can easily be changed without editing your source code. You have to use the Restart button to apply the new values you entered. These values will also be stored inside a Jinx! scene. So you can use a self-programmed effect in various ways driven by only one script file. You will find more about the config variables inside the Jinx!Script Reference chapter.

The last section inside the config screen will give you the state of the byte compiler and the virtual machine. If any errors occurred on compiling or running you will get the messages inside the last error label. If any error occur the script execution will be stopped. If your script takes a too long time to execute, for example when there is an endless loop inside, a timeout will take place and stop your script to prevent any application crashes. In this case you will get an timeout error message in the last error label.

## Jinx!Script Reference

You can use Jinx!Script to create your own matrix effects. It is a very small but powerful programming language which uses an easy to read and learn BASIC like dialect. You have all necessary functions and commands like loops, conditions and even simple sub routines to clearly code your ideas.

### Introduction

Let's take a look at a small sample code:

```
:init
    radius=matrix_y/2
end
:render
    circle matrix_x/2, matrix_y/2, radius, 255, 0, 0, 1
end
```

You can save this code into a text file (we use \*.jns as file extension) and load it with the Jinx!Script engine and you will see a red circle in the middle of your matrix.

As you can see, there are 2 subroutines inside this small code, the first one begins with the label `:init` and the second one with `:render`. The sections are terminated with the `end` command. The section `init` will be called only once by Jinx! on initializing the effect, you can use this section to set up your init parameters or do initial calculations you only need at the start of the effect.

The subroutine starting with the label `:render` will be the main frame generation. This routine is called by Jinx! for every new frame.

There is also an additional close routine which starts with the label `:close`. This is reserved for future use, and will be called when the effect gets stopped. Inside the `:init` and `:close` routines you can't use any drawing commands. They will get ignored because the Jinx!Script engine didn't have any access to the framebuffer at this stage of the program.

To get a feeling what can be done with Jinx!Script and to learn how to program with it, take a look at the sample scripts that are distributed with Jinx!. They will help you to get a fast start into custom effect programming with Jinx!Script.

## Variables

In Jinx!Script you don't distinguish between different variable types. There are only numeric variables and they will be treated as a float. You also don't need to define or dim any variable, this will be done on the fly by Jinx!Script. The variable names can be chosen freely, but must not exceed the 30 character length limit.

To set a variable to a value you simply can use the = operand to assign it, for example:

```
radius=5
```

This will create the variable radius and store the 5 inside. If you use a variable in an expression the first time without any start value it will be created with a start value of 0. All variables in Jinx! are global.

## Arrays

You can also use single dimension array variables. To use a variable as array you simply have to use the index identifier inside square brackets directly after the name, for example:

```
pos_x[0]=10
```

This will set index0 of the pos\_x variable to the value 10. All array variables get created by Jinx!Script on the fly. The index identifier can also be another variable or an expression, even nested array variables are possible, for example something like that:

```
pos_x[ball[3]*2+(4-2)]=10
```

## Expressions and Math functions

With expression you can do calculations with the following operators:

+	addition
-	subtraction
*	multiplication
/	division
^	to raise to the power

The mathematical order the operators will be processed will be ^ -> \*, / -> +, -

To influence these order you can also group your terms with round brackets, for example:

```
pos_x=(17+4)*3/(2+5)
```

Additionally you can use the following functions:

sin(x)	get the sine from x
cos(x)	get the cosine from x
sqr(x)	get the square root from x
rnd(x)	get a random number between 0 and x

If you use the random generator you have to know that it will only give you integer values, so if you need a random value between 0 and 1, you have to do something like:

```
random_number=rnd(100)/100
```

## Relational and Logical Operators

To perform condition checks with `if` or `while` statements the following conditional operators are available:

```
=      equal to
<      less than
<=     less than or equal to
>      greater than
>=     greater than or equal to
<>     not equal to
```

You can also connect multiple conditions with the following logical operators:

```
&      AND, all conditions must be TRUE
|      OR, only one condition has to be true
```

The logical AND will be processed first, but you can also group the conditions with round brackets. For example:

```
if (a+b < c+d | e+f > g+h) & i=j
```

Here the logical OR will be processed first.

## Labels, Subroutines and Jumps

Labels inside Jinx!Script can be used to start a subroutine or simply as a position marker you can easily jump to with the `goto` command.

Labels are started with a colon and will look like:

```
:init
```

There are 3 reserved labels which has special meanings in Jinx!Script:

```
:init      this routine will be started once when the program initializes
:render    this routine will be invoked for every new frame Jinx! has to generate
:close     this one will be started once when the effect will be stopped
```

The `:render` routine must exist as main program, the `:init` and `:close` routines are optional. The `:close` routine is reserved for future use. It doesn't make any sense to use it at the moment, because there is nothing that has to be uninitialized when we stop the engine.

All 3 labels also has to be terminated with the `end` command.

## Jinx! – LED Matrix Control

You can create labels at any place inside your source code and jump directly to that position with the `goto` command, for example:

```
x=1

if x=1
    goto jumppoint
endif

y=2

:jumppoint

z=3
```

Here the program will directly go over to `z=3` and will not process the `y=2` statement.

You can also use a label to start a subroutine. You can call a subroutine with the `gosub` command and will return to the previous point with `return`, nested subroutines are also supported. For example:

```
:render

    gosub new_values
    pset x, y, 255, 0, 0

:end

:new_values

    x=rnd(matrix_x-1)
    y=rnd(matrix_y-1)

return
```

## If / Else Conditions

To make decisions inside your source code you have to use the `if` command. The `if` command will check if a condition is true or false and will only run the included code if the state is true. For example:

```
if x=0
    x=matrix_x-1
endif
```

If `x` will have a value of 0 it will be set to the last horizontal point of your matrix. If `x` has any other value the expression `x=matrix_x-1` will get ignored. You can also use the `else` command to additionally do something if the condition was false:

```
if x=0
    x=matrix_x-1
else
    x=0
endif
```

If you have to check multiple values you can also use `else` and `if` in combination like this one:

```
if x=0
    y=1
else if x=1
    y=2
else if x=2
    y=3
else
    y=4
endif
```

Now we will check the variable `x` for various values. If the value is 0 the variable `y` will be set to 1, if its 1 we will set `y` to 2 and so on. If `x` has any other value than 0,1 or 2 `y` will be set to 4. The `if` and `else if` condition can be a combination of multiple conditions like already pointed out in the Relational and Logical Operators chapter.

### While / Wend Loop

For a conditional loop you can use the `while` and `wend` commands. The `while` command will start a loop and will stay inside the loop as long as the `while` condition is true. The end of the loop section will be marked with the `wend` command. For example:

```
x=0
while x <> 3
    x=x+1
wend
```

The code inside the `while/wend` statements will get executed as long as `x` doesn't has the value 3, as soon as the value has reached 3 the program will continue after the `wend` command.

The condition can also be a combination of multiple conditions, see the chapter Relational and Logical Operators for more info.

### For / Next Loop

Even if it's possible to use a `while/wend` loop for counting purposes it is more useful to take the `for/next` loop instead. The `for/next` loop will take a count variable with a start value and counts up, or down, until the given end state is reached.

```
for n=1 to 10
    y=n
next
```

The loop is started with the `for` command, followed by the count variable and the start and end values. In this example the count variable `n` will start at 1 and count up by 1 on every iteration until it reached the value 10. If the start value is greater than the end value, the `for` loop will count backwards with -1. The end of the loop section is marked with the `next` command.

## Jinx! – LED Matrix Control

Additionally you can use the `step` parameter to the `for` command to adjust the counting step by yourself:

```
for n=1 to 100 step 10
    y=n
next
```

This will start with 1 and will increase the value on every iteration by 10 until `n` is greater or equal to 100. You can also set the `step` parameter to a negative value if your start value is greater than the end value.

## Controlling Loops with Break / Continue

There are 2 additional commands to further control a `while/wend` or `for/next` loop. If you reach any condition inside your loop where you want to exit the loop, even if the `while` condition is still true or the count variable on a `for` loop didn't reached the end value yet, you can use the `break` command. The `break` command will immediately leave the loop and will continue the program directly after the `wend/next` command. The count variable in a `for` loop will stay at the actual value.

```
for n=1 to 100
    if n=10
        break
    endif
next
```

In this example we will exit the loop when the count variable `n` has reached the value 10. If you don't want to exit your loop, but have a condition you want directly go to the next iteration, you can use the `continue` command.

```
while n<10
    n=n+1
    if n=5
        continue
    endif
    y=y+1
wend
```

Here the value `y` will not get counted up when `n` reached the value 5. If `n` reaches the value 5 the `continue` command will directly jump back to the `while` command without executing the other commands before the `wend` command.

## The Pset command

To draw a pixel onto your matrix you will have to use the `pset` command:

```
pset x, y, red, green, blue
```

This will draw a single pixel at the position `x, y` with the `rgb` color values `red, green` and `blue`. The screen orientation inside matrix starts at the upper/left pixel with 0,0. For example:

```
pset 0, 0, 255, 0, 0
```

will draw a single red pixel in the upper, left corner of your matrix.

## The Pget Command

With the `pget` command you can read the actual color of an already drawn pixel:

```
pget x, y, var_red, var_green, var_blue
```

This will take the `rgb` values of the pixel `x, y` and will store it into the variables `var_red`, `var_green` and `var_red`. This will read the pixel color of the upper left corner and draws a pixel with the same color at the upper, right corner:

```
pget 0, 0, red, green, blue  
pset 1, 0, red, green, blue
```

## The Line Command

To draw a complete line onto your matrix you will use the following syntax:

```
line x_start, y_start, x_end, y_end, red, green, blue
```

This will take the pixel `x_start, y_start` as the starting point and draw a line to the pixel `x_end, y_end` with the `rgb` color values `red, green` and `blue`.

## The Rect Command

You can also directly draw a rectangle as a filled or open shape.

```
rect x_start, y_start, x_end, y_end, red, green, blue [, fill]
```

This will draw a rectangle with `x_start, y_start` as upper/left corner and `y_end, y_end` as lower/right corner with the given `rgb` values. The `fill` parameter is optional. If you don't use it or set it to the value `0` you will get an open shape. If you set the `fill` parameter to anything else than `0` you will get a filled rectangle.

## The Circle Command

To draw an open or filled circle you will need the `circle` command:

```
circle x_origin, y_origin, radius, red, green, blue [, fill]
```

The values `x_origin` and `y_origin` will point to the mid-point of your circle. The `fill` parameter after the color values is optional. If you don't use it or set it to `0` you will get an open shape. For every other value you will get a filled circle.



## The Text command

You can also render text directly onto your matrix. The syntax of the text command looks like that:

```
text x, y, center, height, red, green, blue, "Text", "Font" [, style]
```

The `x`, `y` values will represent the drawing position on your matrix. When the center value is set to 0, the upper/left corner of the text will be placed at the `x/y` position. If the center value is set to anything else, the `x/y` position would be in the center of your text.

The height will be given in pixel and after the color values you can define the text and the font you want to use. Remember that there are no string variables in Jinx!Script, so you have to set the text directly here into the inverted commas. The font parameter must match a font name that is installed on your system, e.g. "Arial". The style value is optional and will be assembled of 3 values:

```
style value = 1      text will be drawn bold
style value = 2      text will be drawn italic
style value = 4      antialiasing will be used
```

You can combine these style values by simply adding it, so you can draw an bold and italic text with antialiasing with the style value 7. If you don't use the style value it will be set to 0 and you get a normal font weight without antialiasing. This will for example draw the word Jinx! at the upper left corner of your matrix in a red color and a bold style with antialiasing and the font Arial:

```
text 0, 0, 0, matrix_y/2, 255, 0, 0, 0, "Jinx!", "Arial", 5
```

## The Clear Command

With the `clear` command you can clear your matrix and set all pixels to black. You simply have to use

```
clear
```

without any additional parameters.

## The Fade command

To get some effects like you know from the simple lines engine, you can fade out your matrix. This means with every new frame the old picture of your matrix will be darkened by a given value until its completely black. When you draw new things on every frame and you fade out the old one, you will get a smooth fading effect.

```
fade fade_value
```

The `fade_value` will be the amount your color values will get decreased. For example a value of 10 will fade out a full color (value 255) in 26 frames.

## The Config Command

You can mark a total amount of 10 variables with the `config` command to appear inside the Jinx!Script engine config. These variables can get adjusted inside Jinx!, means outside the source code, and will get stored inside a scene. With this feature you can use one script in many variations without editing the source code. To export a variable into the config screen you simply use the `config` command:

```
config radius=8
```

This will set the variable `radius` to a startup value of 8 and make it visible in the config window. Attention, when you assign a variable with the `config` command you only can assign an initial value, you can't use another variable or a math expression with this command.

## System Variables

There are a few system variables that can be used:

```
matrix_x      the actual horizontal matrix size
matrix_y      the actual vertical matrix size
pixelstep     the configured pixelstep
pi            simply PI: 3,14159..
```

Additional you can use the autocolor feature of Jinx! if you activate it in the Jinx!Script engine config (see chapter The Jinx! Effect Engines).

```
autocolor_red   red value or 255 if autocolor is inactive
autocolor_green green value or 255 if autocolor is inactive
autocolor_blue  blue value or 255 if autocolor is inactive
```

If you want to use an audio trigger, which has also to be activated and adjusted inside the engine config, you will use the following variable:

```
audio_trigger  1 trigger active, 0 trigger inactive or switched off
```

## Comments inside the Source Code

To get your source code readable you can use comments. A comment line has to start with the symbol `#`. After that you can write down whatever you want. As we are completely line oriented, you can't make any comments after a command, always use a new line for comments:

```
#you can use the hash sign to make comments inside your source code
```

## Additional Coding Rules and Notes

- Jinx!Script is line oriented, only use 1 command in 1 line
- Jinx!Script is not case sensitive write with capital letters or not, it's your decision
- variable names are limited to 30 characters
- all drawing command don't break if you draw outside the matrix
- the color values on all drawing commands will use a clamp and get limited to 0-255, so if you set the value to 1540 it will get limited to 255 or if its -15 it will be set to 0
- all drawing command cannot be used inside the :init and :close routines
- a script will get stopped by a timeout if the execution time is to long

# Jinx!Script Command Chart

## Expressions and Math Functions

+	addition
-	subtraction
*	multiplication
/	division
^	to raise to the power
sin(x)	get the sine from x
cos(x)	get the cosine from x
sqr(x)	get the square root from x
rnd(x)	get a random number between 0 and x

## Relational and Logical Operators

=	equal to
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
<>	not equal to
&	AND, all conditions must be TRUE
	OR, only one condition has to be true

## Labels, Subroutines and Jumps

:label	create a label with the name "label"
goto label	jumps directly to the label "label"
gosub label	jumps directly to "label"
return	returns after a gosub command
end	program end

:init, :render and :close are reserved and must be terminated with end. :init and :close are optional

## If/Else and Loops

If a>b	starts an if section
else	do if condition was false
endif	terminates an if section
else if	else if combination for advanced if structures
while a>b	stay in loop while condition is true
wend	marks the end of the loop
for n=a to b [step c]	start a counting loop with a until b is reached, step is optional
next	terminates the loop
break	exit loop immediately
continue	directly go to next loop iteration

## Graphical Commands

clear	clear matrix / blackout
fade a	fade whole screen down by value a
pset x, y, r, g, b	plot pixel
pget x, y, r, g, b	get pixel color
line xs, ys, xd, yd, r, g, b	
rect x1, y1, x2, y2, r, g, b [,fill]	
circle x, y, radius, r, g, b [,fill]	
text x, y, center, height, r, g, b, "TEXT", "FONT" [, style]	

Set optional fill to a value <> 0 to get a filled shape, the style value for the text command can be assembled by the adding the following values:

style value = 1	text will be drawn bold
style value = 2	text will be drawn italic
style value = 4	antialiasing will be used

## System Variables

matrix_x	horizontal matrix size
matrix_y	vertical matrix size
pixelstep	configured pixelstep
pi	simply PI: 3,14159...
autocolor_red	red value or 255 if autocolor is inactive
autocolor_green	green value or 255 if autocolor is inactive
autocolor_blue	blue value or 255 if autocolor is inactive
audio_trigger	1 trigger active, 0 if inactive or off

## Comments

#you can use the hash sign to make comments inside your source code