PixelController
LED Matrix Software

Anleitung v1.0, Januar 2013

Inhalt

Dieses Handbuch wurde mit größter Sorgfalt und dem Gedanken daran erstellt, dass es Ihnen, dem Anwender von PixelController, seien Sie nun Computer-Profi, eifriger DIY-Artist oder blutiger Anfänger, eine sinnvolle Hilfe darstellt. Daher finden sich in ihm unter Umständen Hinweise, die Menschen mit Erfahrung überflüssig erscheinen. Weil PixelController allerdings möglichst vielen Menschen zugänglich sein soll, haben die Autoren hier und da auch auf offensichtliches hingewiesen.

Grundsätzlich ist es unwahrscheinlich, dass durch die Verwendung der Software PixelController und/oder dieses Handbuches Schäden an Ihrem Gerät entstehen kann.
Dennoch weisen wir darauf hin, dass für evtl. entstehende Schäden an Ihrer Hardware oder Ihrem Computersystem keine Haftung übernommen wird.

Die Autoren wünschen eine kurzweilige Zeit beim Studium des Handbuches und viel Erfolg bei der Arbeit mit PixelController.

Michael Vogt	Dr. Stahl
michu@neophob.com	drstahl@gmx.de

In diesem Handbuch werden Sie hin und wieder einige grafische Symbole und besonders formatierten Text finden. Sie sollen Ihnen den Umgang mit PixelController erleichtern, indem sie Sie auf wichtige Punkte zur Beachtung hinweisen, oder Ihnen helfen, Sachverhalte im Umgang mit der Hard- und Software besser zu verstehen.

ACHTUNG!	Diesen Stellen im Handbuch sollten Sie besondere Aufmerksamkeit zukommen lassen. Hier erhalten Sie Hinweise, die Fehler zu vermeiden helfen.
Hinweis	Ein Hinweis wird Ihnen zeigen, was zu tun ist.
Beispiel	Beispiele dienen zur Erläuterung von Sachzusammenhängen.
`text`	Diese Schrift sehen Sie, wenn eine Terminal-Ausgabe simuliert wird oder ein Ausschnitt eines Textfiles angezeigt wird.

In diesem Handbuch werden Sie an einigen Stellen auf Markenamen treffen, die hier lediglich zum Zwecke der Verständlichkeit genannt werden. Weder der Hersteller der PixelController-Software, noch die Verfasser dieses Handbuchs machen sich durch die Nennung der Markennamen diese, oder die Produkte der Markeninhaber zu Eigen.

Einführung

PixelController?

PixelController ist eine flexible Software zum Ansteuern von LED Matrizen. Dabei unterstützt PixelController verschiedene Hardware und ermöglicht eine flexible Steuerung der Applikation. Neben einem GUI stehen verschiedene Schnittstellen zur Verfügung.

Webseite: http://www.pixelinvaders.ch
Kontakt: michu@neophob.com

Unter http://code.google.com/p/pixelcontroller/downloads/list kann PixelController heruntergeladen werden.

Source Code / Lizenz

Die Software wurde unter der GPLv3 OpenSource Lizenz veröffentlicht. Die Lizenz und der Source Code sind unter https://github.com/neophob/PixelController einsehbar.

Unterstütze Betriebssysteme

PixelController ist in Java geschrieben und läuft daher unter Windows©, OSX© und Linux.

Unterstütze Hardware

PixelController wurde ursprünglich geschrieben, damit PixelInvaders 3D-Panel (http://pixelinvaders.ch/) gesteuert werden können. Mittlerweile unterstützt PixelController auch andere Hardware:

Name	Infos
PixelInvaders 3D Panels Arduino/Teensy Firmware vorhanden	80cm x 80cm große DIY 3D Panels, siehe http://www.pixelinvaders.ch. Mehrere Panels können zusammengeschlossen werden.
Seeedstudios Rainbowduino V2 Arduino/Teensy Firmware vorhanden	Produktion eingestellt.
Seeedstudios Rainbowduino V3 Arduino/Teensy Firmware vorhanden	Kleine und handliche 8x8 RGB LED Matrix driver, siehe http://www.seeedstudio.com/wiki/Rainbowduino_v3.0
ArtNet Devices	Unterstützt werden mehrere Universen mit bis zu 510 Kanälen (170 RGB-Pixel pro Universe) Siehe http://www.artisticlicence.com/
MiniDmx Devices	MiniDMX ist ein offener Standard, unterstützt z.B. das SEDU Board von http://www.led-studien.de
Generic UDP Devices	Kann z.B. mit der PixelPi Software auf einem Raspberry Pi verwendet werden. Siehe https://github.com/scottjgibson/PixelPi
TPM2 Serial devices	Offener Standard von http://ledstyles.de Details: http://www.ledstyles.de/fpost296621.html
TPM2 Net devices	Offener Standard von http://ledstyles.de Details: http://www.ledstyles.de/fpost296621.html
Element Labs Stealth v2 LED panel Arduino/Teensy Firmware vorhanden	Werden nicht länger produziert. http://cled.barcousa.com/support.html

Installation

PixelController kann bei Google Code (http://code.google.com/p/pixelcontroller/downloads/list) heruntergeladen werden. Das heruntergeladene Zip-File muss nur extrahiert werden, eine Installation ist nicht nötig.

> **Hinweis:**
> Um die Zip-Datei zu entpacken, benötigen Sie eine entsprechende Software. Falls diese nicht Bestandteil Ihres Betriebssystems ist, können Sie z.B. “Winzip” (http://winzip.de) oder “Winrar” (www.winrar.de) herunter laden.

Sie können PixelController jetzt starten und die Software testen - eine Hardware wird nicht benötigt.
Zur Konfiguration von PixelController mit einer Hardware siehe Kapitel “Konfiguration”.

PixelController wird mit einem **Doppelklick** auf das File `PixelController.jar` gestartet.

PixelController für Linux

Falls Sie PixelController unter Linux auf einer anderen Hardware als 32bit Intel© und mit einem seriellen Interface verwenden, müssen Sie geeignete Treiber für die serielle Schnittstelle selbst kompilieren.
Dazu laden Sie zuerst den Source Code von librxtx herunter:
https://github.com/neophob/librxtx/archive/v2.2.tar.gz und kompilieren das Projekt selbst:

```
# ./configure
# make
```

Kopieren Sie anschließend das File `librxtxSerial.so` in das `PixelController/lib` Verzeichnis und starten Sie PixelController neu.

> **Hinweis:**
> Um zu erfahren, wie Sie selbst eine Applikation unter Linux kompilieren, lesen Sie bitte folgenden Link: http://de.wikihow.com/Wie-man-in-Linux-ein-Programm-kompiliert.

Konfiguration

PixelController wird mit dem Properties-File `config.properties` konfiguriert. Diese Datei finden Sie, wie alle Datendateien, im Verzeichnis `Data`.
Öffnen Sie die Datei `config.properties` **mit einem Texteditor** und nehmen Sie zunächst die Einstellungen vor, die PixelController an Ihr System anpassen.

> **Hinweis:**
> **Öffnen** Sie die Konfigurationsdateien **nicht** mit Programmen wie "**Windows Word**" oder "**Open Office Writer**", oder ähnlichen. Solche Programme fügen dem Text Formatierungen bei, die von der PixelController-Software nicht interpretiert werden können.

> **Hinweis:**
> In den **Konfigurationsdateien** werden Sie Zeilen finden, die mit dem Zeichen "**#**" beginnen.
> Dieses Zeichen (Raute, engl. hash) leitet stets einen **Kommentar** ein. In der selben Zeile stehende Zeichen dahinter werden nicht berücksichtigt.

PixelController-Konfiguration

Der Inhalt der Konfigurationsdatei wird in Folge Punkt für Punkt, von oben nach unten, beschrieben.
Sie erhalten nützliche Anweisungen und Hinweise, die Ihnen bei der Errichtung Ihres Systems aus Rechner, Interface zur LED Matrix (z.B. ein Arduino Board) und LED-Panels helfen werden.

Hier sind die **Voreinstellungen für Generatoren** definiert. Generatoren erzeugen den Bildinhalt, für weiter Informationen siehe Abschnitt *Begriffserklärung*.

```
initial.image.simple=logo.gif
initial.blinken=torus.bml
initial.text=PIXELINVADERS
font.filename=04B_03__.TTF
font.size=82
```

> **Beispiel:**
> Möchten Sie, dass nach dem Start von PixelController ein durch Sie erstelltes Bild oder Logo auf Ihrem Panel angezeigt wird, gehen Sie bitte wie folgt vor:
> Speichern Sie das gewünschte Bild (beispiel.jpg) im Verzeichnis `data\pics` ab und definieren Sie dieses Bild als Startbild `(initial.image.simple=beispiel.jpg).`

PixelController verfügt über einen "**ScreenCapture**"-Generator, welcher einen Teil des Bildschirms aufnehmen und in PixelController verarbeitet werden kann. So kann z.B. ein YouTube-Video auf den Panels dargestellt werden kann.

```
#x/y offset for screen capturing generator
#if you define screen.capture.window.size.x as 0, the screen
#capture generator will bedisabled
screen.capture.offset=100
screen.capture.window.size.x=0
screen.capture.window.size.y=300
```

ACHTUNG!
Der “ScreenCapture”-Generator ist Initial **deaktiviert**, kann aber mit der Definition der `screen.capture.window.size.x` Einstellung größer als 0 aktiviert werden.

Beispiel:
`screen.capture.window.size.x=500` aktiviert den “ScreenCapture”-Generator mit einer Breite von 500 Bildschirmpixeln

In der folgenden Sektion werden die **Netzwerkeinstellungen** von PixelController definiert.
Die `net.listening.port`-und `net.listening.addr`-Einstellungen werden vom Command-Line-Interface verwendet. Die `osc.listening.port`-Einstellung definiert den Port, der für die OSC-Kommunikation verwendet wird.
(siehe Kapitel Interfaces/Fernsteuerung)

```
#network port config
#fudi protocol config, used to communicate with pure data sketch
net.listening.port=3448
net.listening.addr=127.0.0.1
net.send.port=3449
#osc protocol config
osc.listening.port=9876
```

Wie oft soll PixelController den **Bildinhalt aktualisieren**? Dies wird mit der Einstellung fps (Frames per Second) definiert:
`fps=20`

Hinweis:
Der die Framerate Ihres Systems limitierende Faktor kann das Interface zur LED-Matrix sein, also z.B. der Arduino-Controller. Wenn das Interface nicht mehr als z.B. 15 Frames pro Sekunde übertragen kann, wird auch die Ausgabe auf Ihrem Bildschirm nur 15 Frames pro Sekunde anzeigen.
Hierzu finden Sie weitere Informationen im Kapitel ANHANG B

Soll das **GUI-Fenster angezeigt** werden, und wenn ja, wie breit darf es maximal sein?
`show.debug.window=true`
Das GUI-Fenster nicht anzuzeigen, macht nur dann Sinn, wenn PixelController komplett via OSC oder CLI gesteuert wird.

Die Einstellung für die **maximale Breite des GUI-Fensters** kommt erst dann zum Tragen, wenn mehrere Panels von PixelController gesteuert werden und definiert die maximal Breite aller Visuals im GUI Fensters.
`maximal.debug.window.xsize=800`

Per default stellt PixelController ein Visual mehr zur Verfügung als konfigurierte Panels vorhanden sind. Werden **mehr Visuals benötigt** (z.B. ein physikalisches Panel und 3 Visuals) können diese hier definiert werden.
`additional.visual.screens=0`

PixelController zeigt im Output-Fenster eine simulierte LED-Matrix. Die Größe eines solchen **Output-Fenster-Pixels** wird hier definiert und beeinflusst so die Größe des gesamten Output-Fensters.
`led.pixel.size=20`

Soll PixelController per default im **Random-Mode** (Zufallsmodus) gestartet werden oder im Manual Mode?
`startup.in.randommode=false`

> **Hinweis:**
> Möchten Sie, dass PixelController im Zufallsmodus startet, ersetzen Sie "`false`" durch "`true`"

PixelController kann angewiesen werden, beim Start ein **spezifisches Preset** (abgespeicherte Voreinstellung) zu laden. Sie müssen das Preset vorher abgespeichert haben, bevor Sie es verwenden können!
`startup.load.preset.nr=0`

> **ACHTUNG!**
> Wird beim Start von PixelController ein Preset geladen, werden die definierten Voreinstellungen für Generatoren überschrieben.

PixelController verfügt über einen integrierten "**Beat Listener**", welcher Kick-, Snare- und HiHat-Audiosignale erkennt. Wenn der Beat Listener aktiv ist, werden Generatoren und Effekte von den Audiosignalen beeinflusst. So werden z.B. bei einem Kick-Audiosignal die Generatoren stärker aktualisiert, als wenn kein Audiosignal vorhanden ist.
`update.generators.by.sound=true`

PixelController Output-Konfiguration

In der Datei `config.properties` wird ebenfalls die verwendete Hardware (die LED Matrix resp. dessen Interface) definiert. Diese Einstellungen finden sie nach folgender Textmarke:

```
### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ###
# OUTPUT SETTINGS #
# enable only ONE output device
# (PixelInvaders, Rainbowduino, Art-Net, TPM2, UDP, Adafruit or Minidmx)
### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ###
```

Die **Konfiguration** nehmen Sie bitte wie folgt vor:
Stellen Sie sicher, dass Ihre verwendete Hardware unterstützt wird und konfigurieren Sie Ihr Output Device wie in diesem Kapitel erklärt. Wenn Sie Ihre Hardware konfiguriert haben, deaktivieren Sie den Null Output indem sie die zwei `nulloutput.devices` Zeilen mit einem # Symbol auskommentieren.

> **ACHTUNG!**
> PixelController unterstützt **einen Output**. Konfigurieren Sie mehrere Outputs, wird PixelController nicht starten.

Generelle Output Einstellungen

Es gibt **generelle Einstellungen** die für alle (oder fast alle) Outputs gelten. Diese generellen Einstellungen werden hier beschrieben.

Je nach Panel oder Leuchtmittel stimmt die **Farbreihenfolge** nicht. Bauart bedingt sind manche LED-Pixel nicht immer so verschaltet, dass die Ansteuerung in der Reihenfolge RGB möglich ist. So ist bei einigen LED-Clustern Rot und Grün vertauscht. Diese kann mit folgender Einstellung definiert werden:
`panel.color.order=RGB`

> **Hinweis**
> Werden mehrere Panels verwendet, kann für jedes Panel die entsprechende Farbreihenfolge angegeben werden. Per Default wird die Farbreihenfolge RGB angenommen. Mögliche Werte sind: RGB, RBG, BRG, BGR, GBR, GRB. Wobei R steht für Rot, G für Grün und B für Blau.

> **Beispiel**
> Bei **zwei Panels:** `panel.color.order=BGR, BGR`

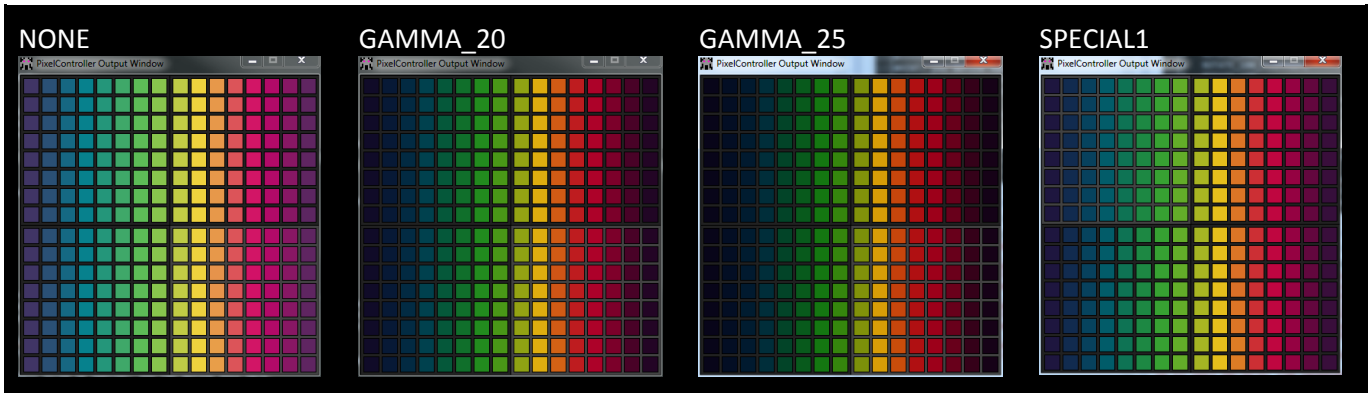
Um die Helligkeit der LED Matrix dem menschlichen Auge anzugleichen, wird die Gammakorrektur angewendet. PixelController kennt vier verschiedene Korrekturfunktionen: *NONE* (keine Korrektur), *GAMMA_20* (Korrektur 2.0), *GAMMA_25* (Korrektur 2.5) und *SPECIAL1* (manuelle Korrektur).
Welche Art der Gammakorrektur für Sie am besten ist, kommt auf die eingesetzten LED Module und den persönlichen Geschmack an.
Die Gammakorrektur wird folgendermaßen definiert :
`panel.gamma.tab=SPECIAL1`

Hinweis

Verwenden Sie **LPD6803**-Module, empfiehlt es sich, die **GAMMA_20**-Korrektur zu verwenden.
Verwenden Sie **WS2801**-basierte Module, verwenden Sie am besten die **GAMMA_25**-Korrektur.
Windows- und OSX-Monitore werden standardmäßig mit einem Gammawert von 2.2 betrieben.

Beispiel

Zur Illustration hier die Auswirkung der Gammakorrektur:



Ist die verwendete LED Matrix in **Schlangenlinien** verkabelt worden, kann das hier definiert werden.
`output.snake.cabling=true`

Ist die LED Matrix hingegen sehr speziell verkabelt worden, kann dies hier definiert werden. Für jeden Pixel wird seine Position (Offset) definiert.
`output.mapping=0,1,4,5,2,3...`

ACHTUNG!

Die Zählung der Pixel startet mit der Ziffer 0, nicht mit 1!

Die **Größe des Panels** (Auflösung in Pixel) wird mit folgenden zwei Definitionen gemacht:
`output.resolution.x=8`
`output.resolution.y=8`

PixelInvaders Output

Hier werden PixelInvaders Panels konfiguriert. Diese Panels können selbst zusammengebaut werden und können unter http://shop.pixelinvaders.ch/product/pixelinvaders-diy-basic-pack bezogen werden. Mehrere Panels können beliebig zu einem größeren zusammengefasst werden.

Hinweis

Ein PixelInvaders Panels besteht immer aus 8 x 8 Pixel und ist in Schlangenlinien verkabelt.

Konfiguriert wird die **Anzahl der verwendeten PixelInvaders Panels** in den folgenden zwei Zeilen (im folgenden Beispiel vier Stück):
`pixelinvaders.layout.row1=NO_ROTATE,ROTATE_90_FLIPPEDY`
`pixelinvaders.layout.row2=NO_ROTATE,ROTATE_90_FLIPPEDY`

Beispiele

Jeder Eintrag definiert ein Panel. Mögliche Optionen für diese Einstellung:

`NO_ROTATE`	Verkabelung von oben Links bis unten Rechts
`ROTATE_90`	Das Visual wird um 90° im Uhrzeigersinn gedreht
`ROTATE_90_FLIPPEDY`	Das Visual wird um 90° im UZS gedreht und horizontal gespiegelt
`ROTATE_180`	Das Visual wird um 180° gedreht
`ROTATE_180_FLIPPEDY`	Das Visual wird um 180° gedreht und horizontal gespiegelt
`ROTATE_270`	Das Visual wird um 270° im Uhrzeigersinn gedreht

Sie können die Panels quasi beliebig aufbauen und anschließend das, was dargestellt werden soll, Panel für Panel drehen und spiegeln.

PixelInvaders Panels werden automatisch detektiert. Dazu werden alle seriellen Ports getestet.
Um diesen Prozess zu beschleunigen, können spezifische Ports ausgeklammert werden (Blacklist):

Beispiel

Beispiel für Linux/OSX:
`pixelinvaders.blacklist.devices=/dev/ttyUSB000`

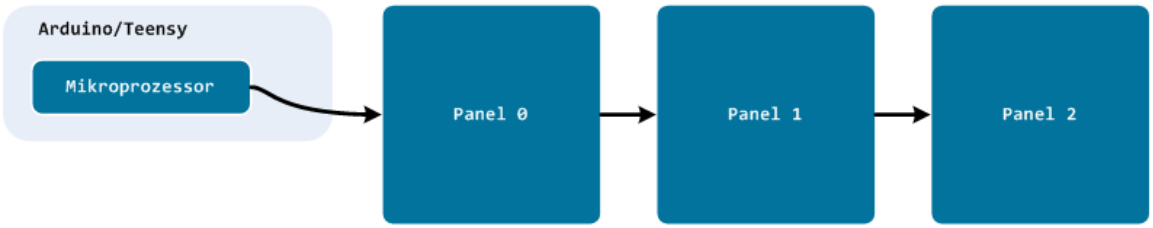
Beispiel

unter Windows soll COM2 nicht getestet werden:
`pixelinvaders.blacklist.devices=COM2`

PixelInvaders kann, wie oben beschrieben, mehrere Panels ansprechen. Sind die Panels horizontal und vertikal angeordnet, muss der **Verkabelungsweg** mit der Anweisung `pixelinvaders.panel.order=0,3,4,1,2,5` Konfiguriert werden.

Beispiel horizontal+

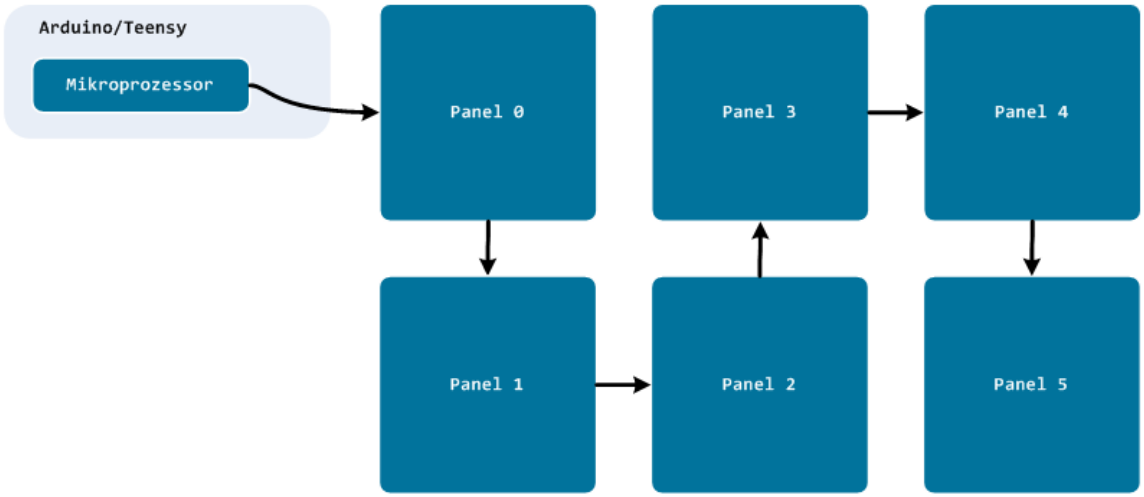
3 Panels sind folgendermaßen verkabelt:



Die Panels sind aufsteigend angeordnet, es muss nichts definiert werden.

Beispiel horizontal und vertikal

Sind die Panels jedoch auch Vertikal angeordnet, muss diese Anordnung definiert werden. Hier eine Installation mit 6 Panels:



Die Konfiguration dieser 6 Panels sieht folgendermaßen aus:

```
pixelinvaders.panel.order=0,3,4,1,2,5
```

ACHTUNG!

Das erste Panel startet bei der Nummer 0.

Das ist das Panel, an welchem das Kabel, von Ihrem Arduino kommend, angeschlossen ist.

Null Output

Der Null-Output definiert ein **Dummy Device,** mit dem PixelController auch ohne Hardware, also ohne Interface und auch ohne LED-Matrix, verwendet werden kann. Dieser Output ist per default definiert.

```
nulloutput.devices.row1=2
nulloutput.devices.row2=2
```

Hinweis

Bitte beachten Sie, die generellen Einstellungen um weitere Parameter (Auflösung der Matrix etc.) zu definieren.

RainbowduinoV2 Output

Die Rainbowduino-Boards müssen die **Neorainbowduino-Firmware** installiert haben, welche unter http://code.google.com/p/neorainbowduino/ heruntergeladen werden kann. Jedem Rainbowduino-Board muss eine eindeutige I2C Adresse zugewiesen werden, welche zur eindeutigen Adressierung verwendet wird. Im folgenden Beispiel werden 4 Rainbowduino Boards mit den I2C Adressen 5,6,8 und 9 angesprochen:

Beispiel

Bei vier Rainbowduino Boards:

```
layout.row1.i2c.addr=5,6
layout.row2.i2c.addr=8,9
```

RainbowduinoV3 Output

Die Rainbowduino-Boards müssen die **rainbowduino-v3-streaming-firmware** installiert haben, welche unter https://code.google.com/p/rainbowduino-v3-streaming-firmware/ heruntergeladen werden kann.
Jedes Rainbowduino Device wird via USB angeschlossen.

```
layout.row1.serial.devices=/dev/ttyUSB0,/dev/ttyUSB1
layout.row2.serial.devices=/dev/ttyUSB2,/dev/ttyUSB3
```

> **Hinweis**
> Bitte beachten Sie die generellen Einstellungen um weitere Parameter (Auflösung der Matrix etc.) zu definieren.

Stealthpanel Output

Die Konfiguration ist Analog der PixelInvaders Panels:

```
stealth.layout.row1=NO_ROTATE
stealth.layout.row2=NO_ROTATE
```

> **Hinweis**
> Bitte beachten Sie die generellen Einstellungen um weitere Parameter (Auflösung der Matrix etc.) zu definieren.

Artnet Output

Der ArtNet Output kann verwendet werden, um mit einem ArtNet-to-DMX-Konverter Daten an ein oder mehrere DMX Universe zu senden. Pro Universe stehen 512 Channels zur Verfügung, somit können maximal 170 RGB Pixels pro Universe angesprochen werden.

```
artnet.ip=192.168.1.2
artnet.pixels.per.universe=170
artnet.first.universe.id=1
```

> **Hinweis**
> Bitte beachten Sie die generellen Einstellungen um weitere Parameter (Auflösung der Matrix etc.) zu definieren.

UDP Output

Dieser Output schickt die Bildinformationen via UDP an ein entfernten Rechner. Dies macht z.B. Sinn mit der PixelPi Software (https://github.com/scottjgibson/PixelPi). PixelPi läuft auf einem Raspberry Pi und kann ein WS2801 Panel ansteuern.

```
udp.ip=192.168.111.25
udp.port=6803
```

> **Hinweis**
> Diese Output Methode bietet keinen Schutz vor Übertragungsfehlern, da keine Header Informationen mitgeschickt werden.

Hinweis
Bitte beachten Sie die generellen Einstellungen um weitere Parameter (Auflösung der Matrix etc.) zu definieren.

TPM2 Serial Output

TPM2 Serial kompatible Kontroller werden mit folgenden Einstellungen konfiguriert:
`tpm2.device=/whatever/youwant`
`tpm2.baudrate=115200`

Beispiel
Beispiel für Linux/OSX:
`tpm2.device=/dev/ttyUSB000`

Beispiel
unter Windows soll COM2 nicht getestet werden:
`tpm2.device=COM2`

Hinweis
Bitte beachten Sie die generellen Einstellungen um weitere Parameter (Auflösung der Matrix etc.) zu definieren.

TPM2 Net Output

Für den Betrieb dieses Devices müssen Sie die IP Adresse definieren:
`tpm2net.ip=192.168.111.25`

Hinweis
Bitte beachten Sie die generellen Einstellungen um weitere Parameter (Auflösung der Matrix etc.) zu definieren.

MiniDMX Output

Die einzige Einstellung für den Betrieb eines MiniDMX-fähigen Devices ist die Definition der seriellen Baudrate:
`minidmx.baudrate=115200`

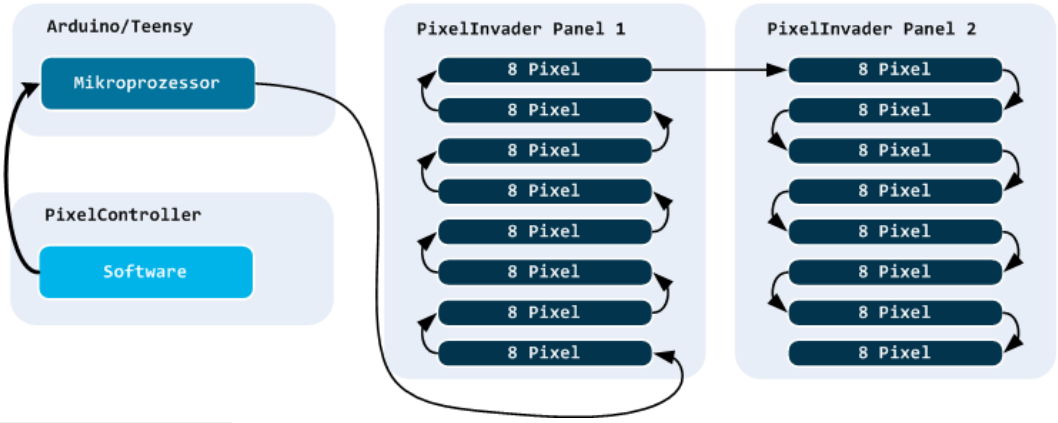
Hinweis
Bitte beachten Sie die generellen Einstellungen, um weitere Parameter (Auflösung der Matrix, etc.) zu definieren.

Beispielkonfigurationen

Zur Illustration werden hier zwei beliebige Beispielkonfigurationen dargestellt.

PixelController Konfiguration

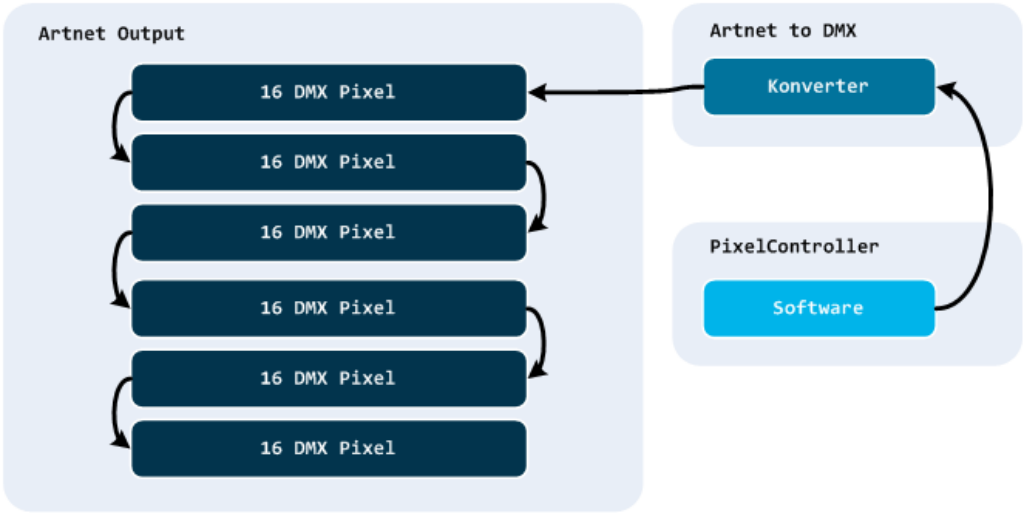
Zwei PixelInvaders Panels, welche unterschiedliche LED Module installiert haben, werden von PixelController angesteuert. Damit die Farben besser zu Geltung kommen, wird eine Gammakorrektur von 2.5 angewendet.




```
panel.color.order=RGB, RBG
panel.gamma.tab=GAMMA_25
pixelinvaders.layout.row1=ROTATE_90_FLIPPEDY, NO_ROTATE
```

ArtNet Konfiguration

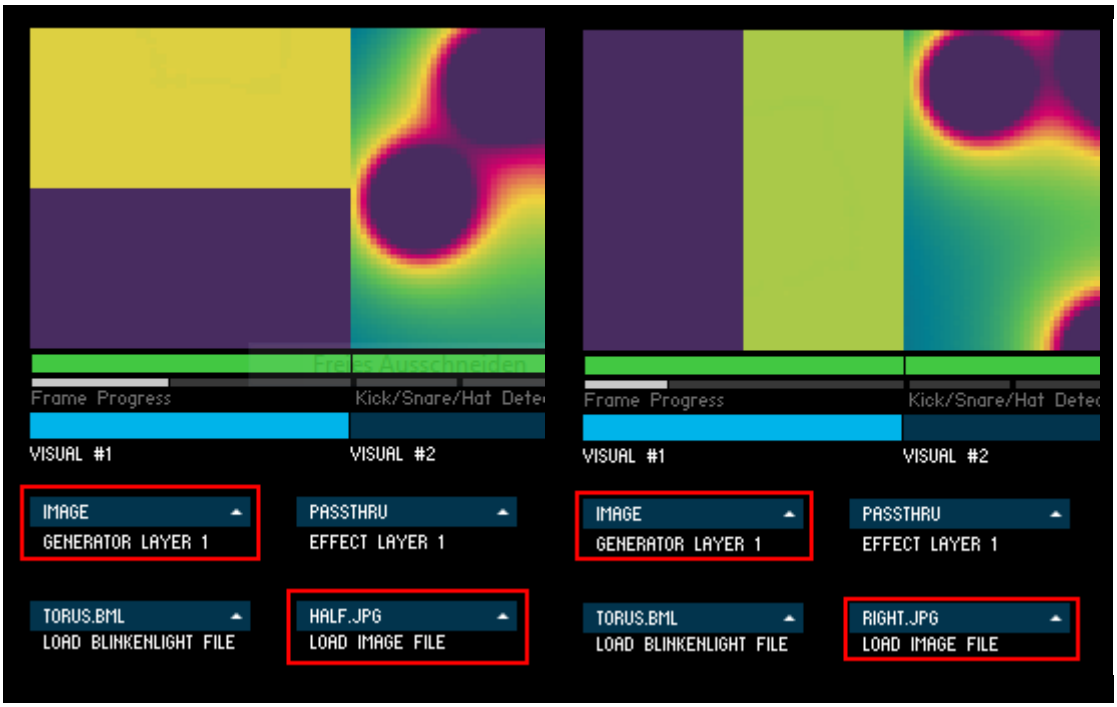
Ein ArtNet Panel mit der Auflösung von 16 x 6 Pixel wird von PixelController angesteuert. Das Panel ist in Schlangenlinien verkabelt. Damit die Farben besser zu Geltung kommen, wird eine Gammakorrektur von 2.5 angewendet.




```
panel.gamma.tab=GAMMA_25
output.snake.cabling=true
output.resolution.x=16
output.resolution.y=6
output.layout=ROTATE_180_FLIPPEDY
artnet.ip=192.168.1.2
artnet.pixels.per.universe=170
artnet.first.universe.id=1
```

Konfigurations-Tipps

Zum Überprüfen der Konfiguration starten Sie PixelController und wählen als “*Generator Layer 1*” aus. Wählen Sie anschließend das Bild “*Half.jpg*” und “*Right.jpg*” aus. So können Sie überprüfen, ob die Panels richtig konfiguriert wurden.



Weitere Konfigurationsdateien

Im Verzeichnis `Data` existieren noch weitere Konfigurationsfiles, welche PixelController benötigt:

- `palette.properties`: In diesem File sind Colorsets abgespeichert. Ein Colorset definiert die Farben, welche PixelController verwendet und kann ganz einfach definiert werden

  ✅ **Beispiel** für das Colorset mit dem Namen Fizz:
  `Fizz=0x588F27, 0x04BFBF, 0xF7E967.`

  Ein Colorset besteht aus einem Namen, hier "Fizz", und mehreren RGB-Farbwerten in Hexadezimal-Notation. Sie können diese Datei mit eigenen Colorsets erweitern, indem Sie einfach weitere Einträge nach obigem Beispiel hinzufügen.
  Es gibt zahlreiche Webseiten, die Sie bei der Erstellung von attraktiven Colorsets unterstützen.

  ✅ **Beispiele:**
  https://kuler.adobe.com/
  http://colorschemedesigner.com/
  http://palette.ws/

- `presents.led`: In diesem File werden die Presets (vordefinierte Animationen) abgespeichert.

  🟥 **ACHTUNG!**
  Manuelle Bearbeitung dieser Datei durch den Anwender ist nicht empfohlen.

- `logging.properties`: In dieser Datei wird definiert, wie PixelController Log-Dateien generiert. WichtigeEinstellungen in diesem File:

  Wie groß darf das Log-File maximal sein (Default 4MB):
  `java.util.logging.FileHandler.limit=4096000`

  Was soll ins Log File geschrieben werden:
  `java.util.logging.FileHandler.level=ALL`

  Was soll in der Konsole ausgegeben werden:
  `java.util.logging.ConsoleHandler.level=INFO`

  Mögliche Log-Level-Einstellungen sind:

ALL	Alle Daten werden mitgeschrieben ("geloggt")
INFO	Nur Info-Daten werden geloggt
WARNING	Nur Warnungen werden geloggt
SEVERE	Nur schwerwiegende Fehler werden geloggt
OFF	kein Logging

Eigene Mediendateien verwenden

PixelController liefert bereits einige Mediendateien mit. Sie können allerdings auch Ihre eigenen Mediadaten benutzen, die Sie z.B. speziell für die Zusammenarbeit mit PixelController angefertigt haben, oder aber auch eigene Daten, die Sie auf Ihrer Festplatte finden (z.B. eigene Urlaubsfotos, Schriftarten, usw.).
Diese Daten können beliebig ausgetauscht und/oder erweitert werden.

Bilder

Bilddaten werden im Verzeichnis `data/pics` abgespeichert und vom "Image"-Generator verwendet. In dieses Verzeichnis können vom Anwender beliebige Bilder abgelegt werden. Die Bilddaten sollten aus Performance-Gründen ein Maß von 512 x 512 Pixeln nicht überschreiten.
Unterstütze Dateitypen: .gif, .jpg, .tga, .png.

> **ACHTUNG!**
> Animierte .gif-Dateien werden nicht unterstützt.

Blinkenlights-Animationen

Blinkenlights[1]-Animationsdateien werden im Verzeichnis `data/blinken` abgespeichert und vom "Blinkenlights" Generator verwendet.

> **Hinweis:**
> Es gibt verschieden Möglichkeiten, Blinkenlights-Animationen zu erstellen:
> - Es gibt ein Processing[2], Plugin[3], mit deren Hilfe Sie Sie selbst Blinkenlights-Animationen aus Processing-Sketches kreieren können.
> - Image to Bml[4] konvertiert eine Bildsequenz in eine Blinkenlights-Animationsdatei.
> - Blinkenligths bietet verschiedene Tools auf seiner Webseite an, mit denen Animationsdateien erzeugt werden können.

TTF-Schrift

Das Schriftfile muss im `data`-Verzeichnis gespeichert sein und wird in der Konfigurationsdatei `config.properties` definiert. Die Konfiguration sieht folgendermaßen aus:

```
font.filename=04B_03__.TTF
font.size=82
```

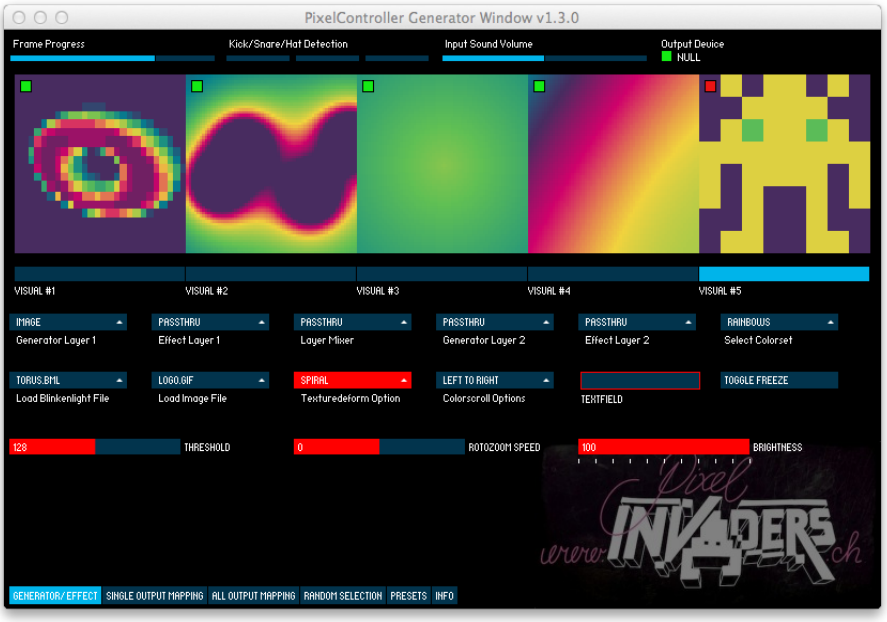
Die TTF-Schrift wird vom "Textwriter"-Generator verwendet.

> **Hinweis:**
> PixelController verarbeitet ausschließlich Schriften im TTF-Format.

[1] http://blinkenlights.net/
[2] http://processing.org/
[3] http://robinsenior.com/blinkenlights/
[4] http://www.capybara.org/~dfraser/archives/261

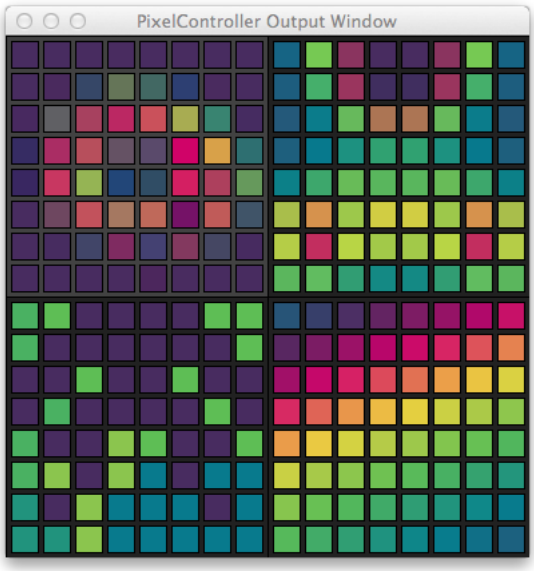
PixelController im Betrieb

Wird PixelController gestartet, werden zwei Fenster geöffnet: das Hauptfenster mit GUI Elementen und das Output Fenster, welches die LED Matrix Ausgabe simuliert.



Dies ist das **Hauptfenster** von PixelController.

Es dient sämtlichen Einstellungen, die unmittelbare Auswirkung auf die Darstellung Ihrer Panels haben.



Das **Output-Fenster** zeigt Ihnen, was aktuell auf den Panels dargestellt wird. Diese Anzeige ist sehr hilfreich, wenn Sie Ihre Panels während deren Betrieb nicht direkt einsehen können, weil Sie sich bspw. hinter einem Bühnenaufbau befinden.

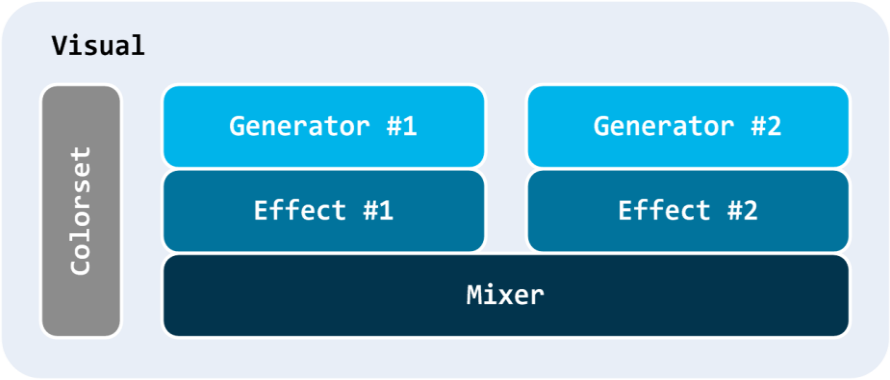
Beispiel:
PixelController steuert 4 Panels mit jeweils unterschiedlichen Visuals.

Begriffserklärung

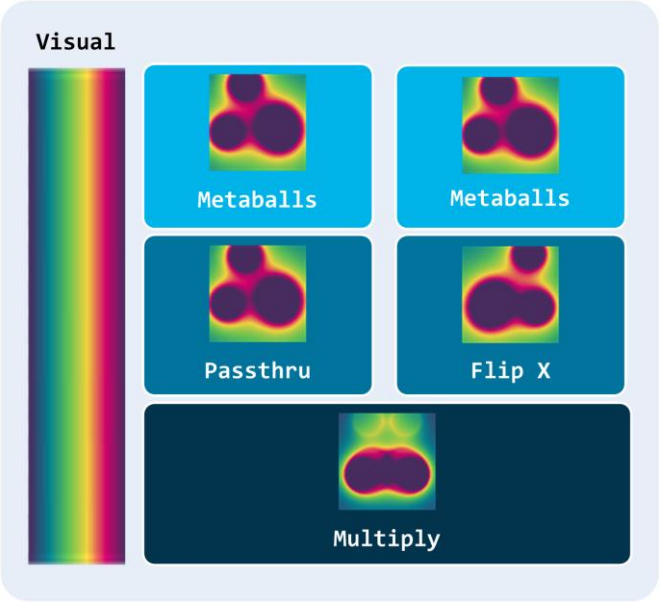
Ein Visual ist eine in Echtzeit generierte Animation und beinhaltet folgende Komponenten:

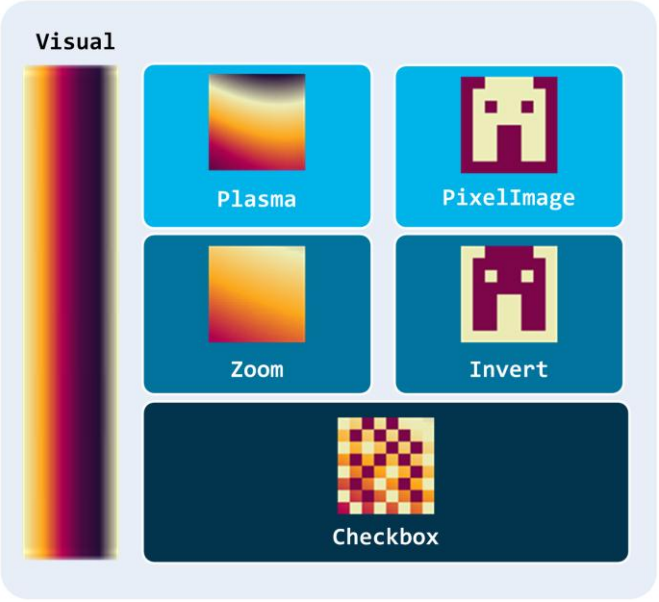
- **Colorset**: definierte die Farbpalette
- **Generator**: generiert den Bildinhalt
- **Effekt**: modifiziert den Bildinhalt
- **Mixer**: führt zwei Bildinhalte zusammen

Grafisch sieht das folgendermaßen aus;

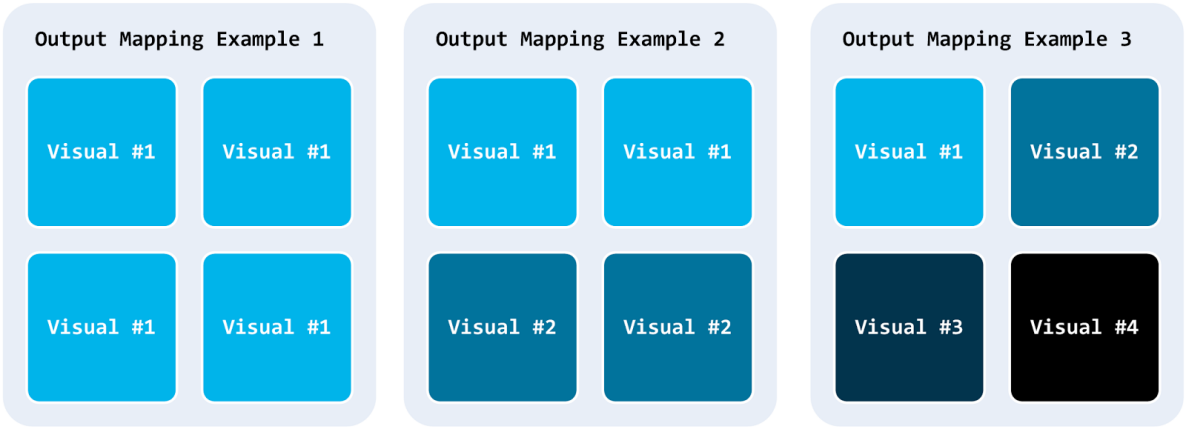


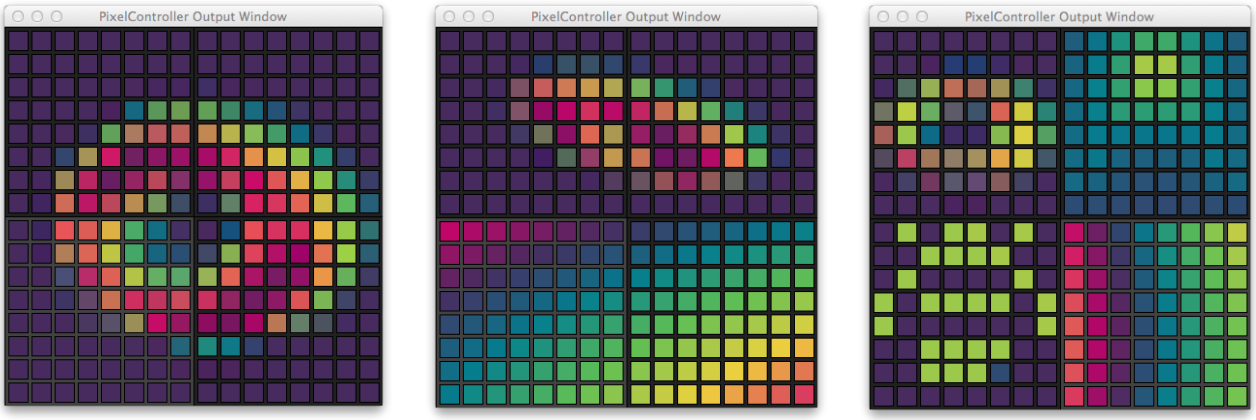
Ein Visual ist die Einheit, welche auf einem oder mehreren Output-Panels ausgegeben wird. Dazu zwei Beispiele:





Pro Output (Physikalisches Panel) wird ein Visual bereitgestellt. Das Visual kann beliebig auf den Output gemappt werden. Hier 3 verschiedene Beispiele für 4 Output-Panels:





✅ **Beispiel Links:**
Alle 4 Panels agieren als ein großes Panel, ein Visual erstreckt sich auf alle 4 Panels.

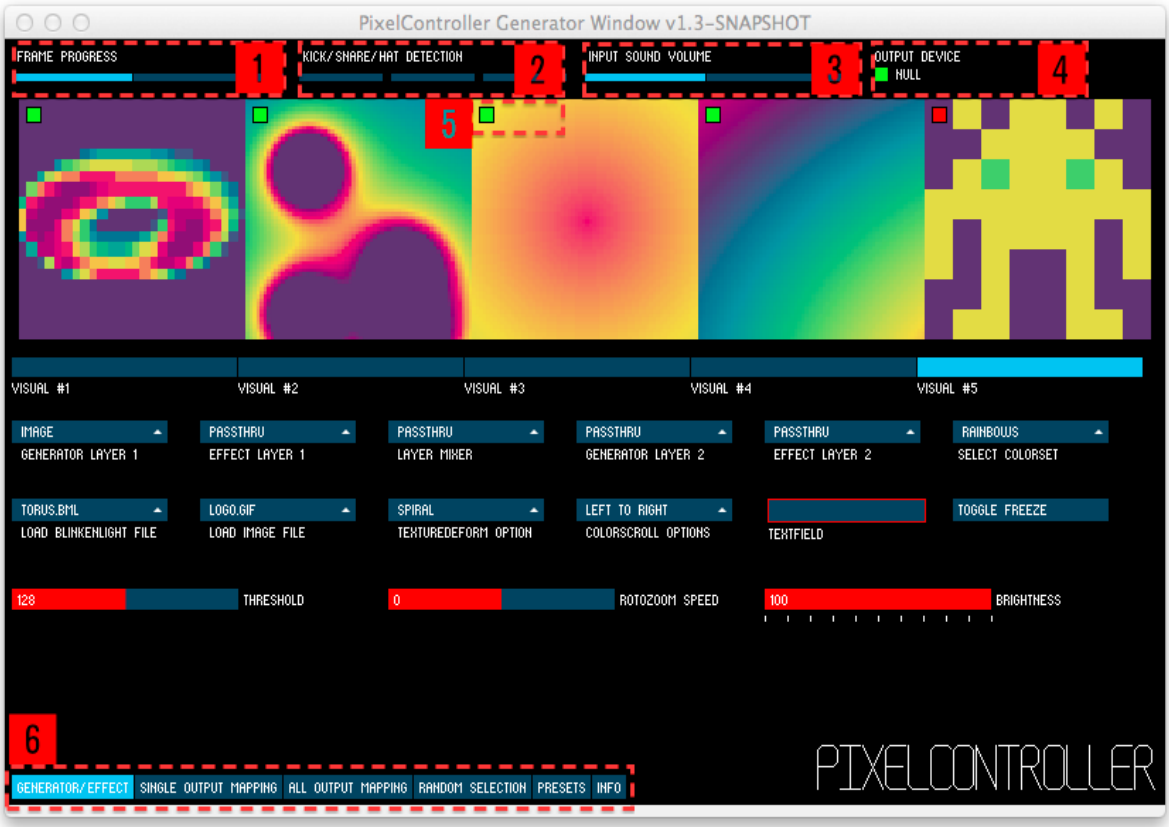
✅ **Beispiel Mitte:**
Die jeweils oberen und unteren Panels agieren als ein Panel.

✅ **Beispiel Rechts:**
Alle 4 Panels agieren selbständig und zeigen ein unterschiedliches Visual.

ℹ️ **Hinweis**
Mehrere Panels werden aktuell von PixelInvaders-, Stealth- und Rainbowduino Panels unterstützt.

PixelController GUI

Das PixelController-GUI-Fenster ist folgendermaßen aufgebaut:



Beschreibung der einzelnen Elemente:

1 **Frame Progress**: Dieser Balken zeigt an, wie schnell Frames aufgebaut werden. Dies ist u.a. abhängig von der Audio-Umgebung.

2 **Kick/Snare/HiHat Detection**: Wenn ein Kick-, Snare- oder HiHat-Audio-Signal erkannt wurde, leuchtet der Indikator auf.

3 **Input Sound Volume**: Zeigt die normalisierte Eingangslautstärke an.

4 **Output Device**: Beschreibt das ausgewählte Output-Device. Unterstützt das Output-Device einen Verbindungsstatus, wird dieser angezeigt.
Arduino Outputs z.B. unterstützen den Verbindungsstatus, UDP-Output-Devices hingegen nicht.

5 **Visual Anzeige**: Zeigt an, ob das Visual auf den Panels angezeigt, bzw. verwendet wird.

6 **Tabulatoren**: Auswahl der verschiedener Funktionen.

Folgende Tabulator-Menus stehen zur Verfügung:

- **Generator/Effekt**: Generatoren, Effekte und Mixer werden definiert.
- **Single Output Mapping**: Hier wird definiert, welches Visual auf welchem Output Panel dargestellt werden soll.
- **All Output Mapping**: Dieses Tab ist nur sichtbar, wenn mehrere Output-Panels vorhanden sind. Hier wird ein Visual auf alle Output Panels gemappt.
- **Random Selection**: Hier wird definiert, welche Optionen im Random Betriebsmodus geändert werden.
- **Presets**: Hier werden Presets abgespeichert und geladen.
- **Info**: Informationen über den Betrieb von PixelController.

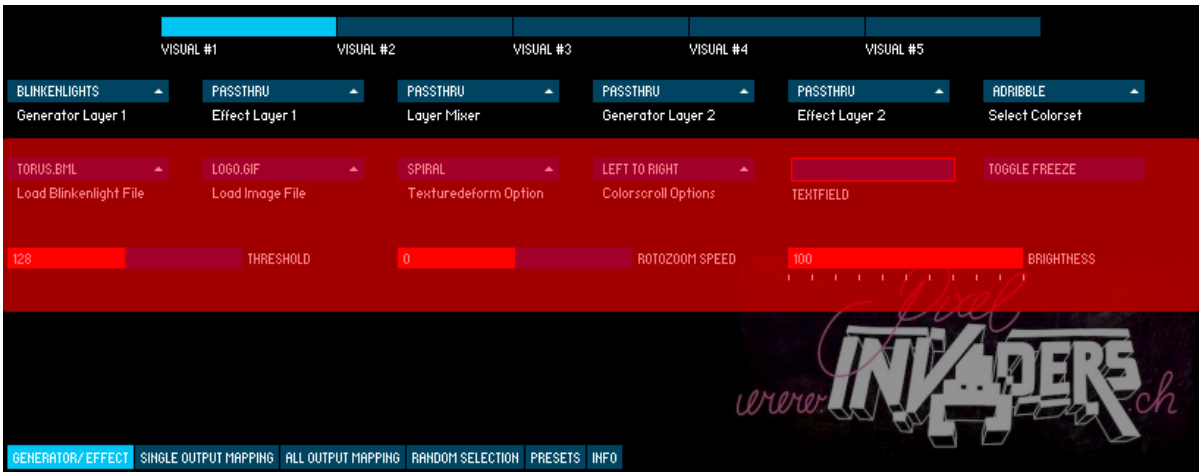
Betriebsmodi

PixelController unterstützt 3 unterschiedliche Betriebsmodi:

- **Manuell** - PixelController wird “von Hand” gesteuert.
  Das kann via PixelController-GUI sein, oder aber auch via MIDI-Device oder OSC-Applikation.
- **Preset** - Vordefinierte, abgespeicherte Einstellungen (engl. Preset) werden geladen.
- **Random** - PixelController wird per Zufallsgenerator gesteuert.
  Dabei kann definiert werden, welche Einstellungen geändert werden.

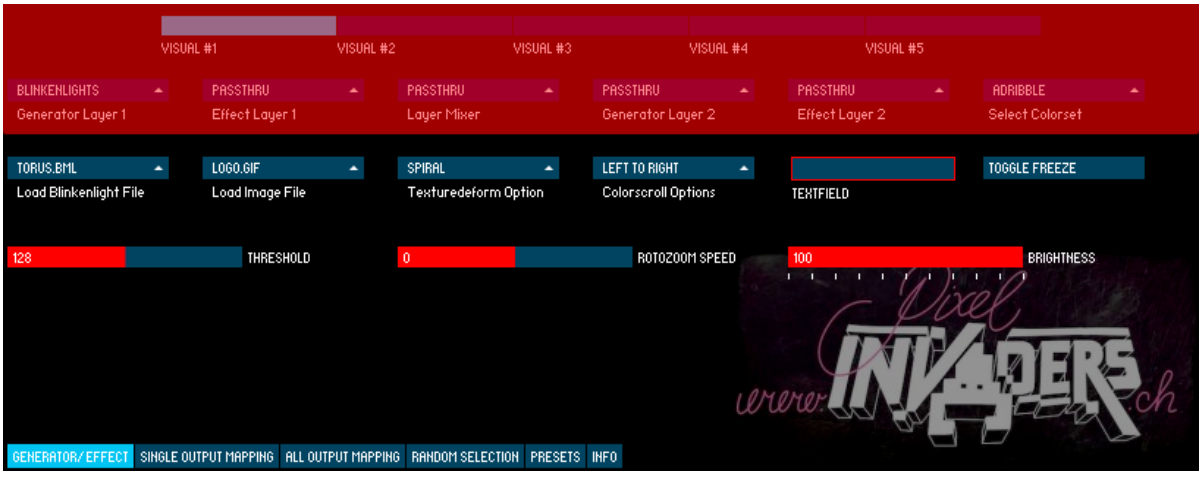
Manuell

Zuerst wird das **Visual ausgewählt**, welches bearbeitet werden soll. Da PixelController immer ein Visual mehr zur Verfügung stellt, als Hardware-Panels zur Verfügung stehen, können Sie Einstellungen an einem Visual vornehmen, ohne das dies bereits auf den Panels sichtbar ist.



In dieser Maske können die **Generatoren** (*Generatoren Layer 1* und *Generatoren Layer 2*), die **Effekte** (*Effect Layer 1* und *Effect Layer 2*), der **Mixer** (*Layer Mixer*) sowie das aktuelle **Colorset** (*Select Colorset*) geändert werden.

Weiter gibt es **globale Einstellungen**, welche sich auf einen Generator oder Effekt beziehen. Hier kann z.B. das Animationsfile (*Load Blinkenlight-File*) geladen werden, oder die Geschwindigkeit des Rotozoomers verändert werden.



Preset

Im *Preset*-Tab können Presets abgespeichert oder geladen werden. Dazu wird eine Preset-Nummer ausgewählt (zwischen 1 und 96), ein Klick auf den *Load Preset*-Button lädt das Preset. Ein Klick auf *Save Preset* speichert die aktuellen Einstellungen im entsprechenden Preset ab. Optional kann für jedes Preset ein Name definiert werden.

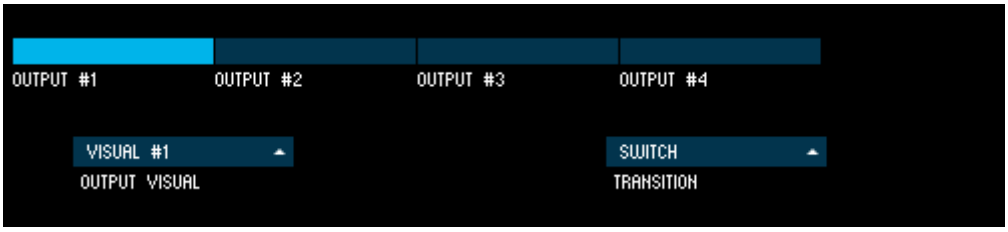
Random

Im *Random Selection*-Tab kann der *Random Mode* aktiviert oder deaktiviert werden. In diesem Modus werden nur die aktivierten Einstellungen (siehe Screenshot unten) modifiziert. Das kann z.B. dazu verwendet werden, dass nur das Color Set geändert wird.
Der Random Mode funktioniert Audio-aktiv, d.h. die meisten Änderungen werden durchgeführt, wenn ein Bass-Kick detektiert wird.

Ein Klick auf den Randomize-Button führt dazu, dass die aktivierten Einstellungen für das aktuelle Visual zufällig erzeugt werden. Das kann zu interessanten Kombinationen führen, welche manuell weiter bearbeitet und abgespeichert werden können.
Ein Klick auf *Randomize Preset* lädt eines vom Anwender zuvor erzeugtes Preset zufällig aus.

Output-Mapping und Übergänge

Visuals können auf einem oder mehreren Output-Panels ausgegeben werden. Deren Konfiguration können Sie im “Single Output Mapping”-Tab vornehmen.



Im Screenshot oben existieren 4 Output Panels. Nachdem das entsprechende Output Panel ausgewählt wurde, können Sie das neue Output Visual auswählen, sowie dessen Übergang (Transition). Folgende Übergänge stehen zur Verfügung:

- **Switch**: Das Visual wird unmittelbar ausgewechselt.
- **Crossfade**: Die Visuals blenden ineinander über. (engl. to fade)
- **SlideUpsideDown**: Das neue Visual schiebt sich von oben nach unten ein.
- **SlideLeftRight**: Das neue Visual schiebt sich von links nach rechts ein.

Tastatur-Shortcuts

Um effektiv mit PixelController arbeiten zu können, werden auch Tastaturbefehle unterstützt:

- **1 bis 9**: Das zu bearbeitende **Visual** wird ausgewählt
- **C**: Auswahl des nächsten **Colorsets**
- **F**: Auswahl des nächsten **Generator 1**
- **G**: Auswahl des nächsten **Generator 2**
- **W**: Auswahl des nächsten **Effekt 1**
- **E**: Auswahl des nächsten **Effekt 2**
- **M**: Auswahl des nächsten **Mixers**
- **R**: **Randomize**, das aktuelle Visual wird per Zufallsmodus zusammengestellt. Dabei werden nur Änderungen vorgenommen, **welche die sichtbaren Visuals nicht beeinflussen**, konkret:
  Wird auf dem Panel z.B. ein Visual mit dem Image-Generator dargestellt, wird das Bild des Image-Generators nicht geändert, das Movie-File des Blinkenlights-Generators jedoch schon da dieses nicht sichtbar ist.

  Hinweis
  Die Buchstaben müssen groß geschrieben werden, d.h., Sie müssen die Shift-Taste gedrückt halten (oder Caps Lock aktiviert haben), um die Tastatur-Shortcuts zu verwenden.

Beat Listener / Audio-Input

PixelController verwendet den Default Soundinput des Rechners, bei Desktop Rechner ist das der LineIn Anschluss, bei Notebooks das interne Mikrofon. Möchte man einen anderen Line-In-Anschluss verwenden, kann das Betriebssystem entsprechend konfiguriert werden.

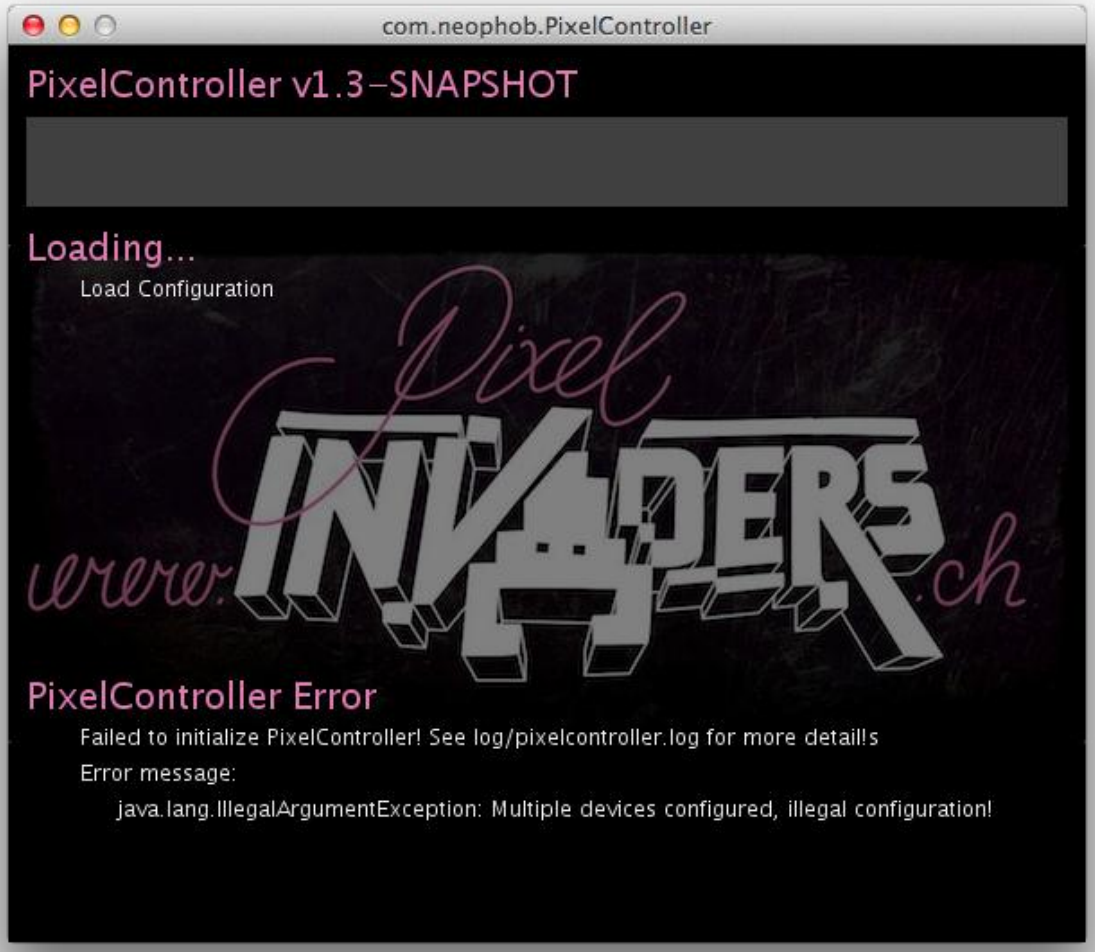
OSX:	Mit dem Tool LineIn kann das Default Input-Device ausgewählt werden.
Windows:	dieser Artikel erklärt, wie man das Default Input-Device wählt.
Linux:	Distributions- und Sound-Layer abhängig (ALSA/OSS), ein Beispiel, wie man das Input-Device für ALSA wählt, finden Sie im Link.

Fehlersuche

PixelController schreibt eine Log-Datei im `Log`-Verzeichnis. Dieses Textfile wird in Fehlerfällen benötigt und soll Ihnen helfen, Fehler zu beheben.

Versuchen Sie, das Log-File zu analysieren und so einen Hinweis auf den Fehler zu finden. Finden Sie keine Lösung für das Problem, können Sie einen Fehlerreport auf der GitHub-Seite von PixelController erstellen. In diesem Fall fügen Sie dem Report immer das Log-File hinzu und beschreiben bitte das Problem so ausführlich wie möglich.

Die Fehlermeldung "*Multiple devices configured, illegal configuration!*" erscheint, wenn mehrere Output-Devices konfiguriert sind:



Hinweis

Wahrscheinlich haben Sie vergessen, das NULL Output-Device auszukommentieren. Fügen Sie in der Konfigurationsdate ein Raute-Zeichen (#) vor die `nulloutput.devices`-Definition ein.

Interfaces / Fernsteuerung

PixelController kann nicht nur durch das GUI bedient werden sondern erlaubt die Einbindung verschiedenster Fremdsysteme, welche hier erklärt werden.

OSC-Schnitstelle

OSC[5] ist ein offener Standard und kann stark vereinfacht als “MIDI via Ethernet” definiert werden. Dank OSC können verschiedene Kontroller verwendet werden um PixelController (Remote) zu bedienen. Populär ist z.B. **TouchOSC** ist eine **iOS** Applikation, welche vom Anwender definierte Layouts unterstützt:

Mit der OSC-Schnittstelle kann man für PixelController ein eigenes Layout erstellen und die Software zB. mit einem iPad bedienen.

Für **Android** gibt es u.U. Control von Charlie Roberts, dafür gibt es im `integration/ControlOSC` Verzeichnis eine Beispielimplementation.

Auch für **Processing** gibt es zwei Beispiel Implementationen im Verzeichnis `integration/Processing`.

Eine andere Idee ist, die PixelController-Software mit Hilfe von **PureData-Extended**[6] über Ihre MIDI-Geräte zu steuern. Siehe dazu das nächste Kapitel.

[5] http://en.wikipedia.org/wiki/Open_Sound_Control

[6] http://puredata.info/

Steuerung von PixelController mit einem MIDI-Gerät

Es ist möglich, mit einem MIDI-Kontroller PixelController zu steuern.
Benötigt werden hierzu:

- Pure-Data Extended (Hier Version 0.43.1-extended-20120614)
- Der Pure-Data Sketch Midi2OSC (`integration/PureData/Midi2OSC.pd`)
- Ein MIDI-Gerät mit Steuerfunktion (MIDI-Keyboard, MPC, Faderbox, etc.)
- (USB) MIDI-Interface (Im Musik-Fachhandel ab ca. 15€ erhältlich)

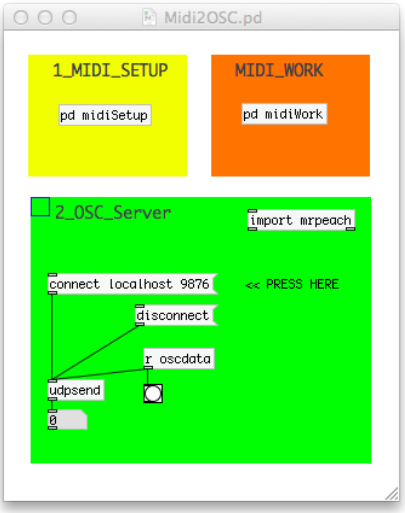
PureData ist eine visuelle Programmiersprache, die unter http://en.flossmanuals.net/pure-data/ detailliert beschrieben ist. PureData verfügt über einen **Editier-** und einen **Ausführungsmodus**. Im Editiermodus können Änderungen am Programm vorgenommen werden, im Ausführungsmodus nicht. Den Editiermodus können Sie in Windows mit Ctrl+E und in OSX mit Cmd+E aktivieren resp. deaktivieren. Alternativ kann dies auch via Edit-Menu gemacht werden.

Vorbedingungen
Stellen Sie sicher, dass Ihr MIDI-Gerät mit Ihrem Rechner verbunden ist und Datentransfer stattfinden kann.

> **Hinweis:**
> Achten Sie darauf, dass der Ausgang des MIDI-Keyboards mit dem Eingang des Interfaces verbunden ist, und verbinden Sie dieses mit Ihrem Rechner.

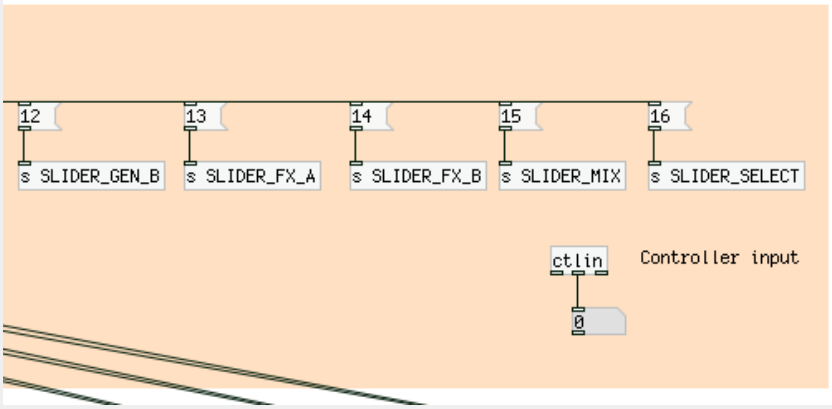
Schritt 1: Starten Sie Pure-Data
Im Menu “Preferences/MIDI Settings” wählen Sie “Media/MIDI Settings” aus.

Wählen Sie anschließend Ihr **Input-Gerät** aus.
(Hier wurde das AKAI MPD26 an Port 1 ausgewählt)



Schritt 2: Converter-Sketch laden
Im Menu “Datei” wählen Sie Laden und wählen die Datei Midi2OSC.pd

Es öffnet sich ein Fenster wie links zu sehen.



Schritt 3: MIDI-Einstellungen
Rechtsklicken Sie auf “pd midi-Setup”, um den MIDI-Einstellungs-Dialog zu öffnen.

Bewegen Sie einen Regler oder Fader, oder drücken Sie eine Taste an Ihrem MIDI-Keyboard, von dem Sie wissen, dass er MIDI-Daten sendet, um seine Controller-ID in Pure-Data angezeigt zu bekommen.
Anschließend können Sie diese ID, den “Controller input”, einem Ereignis zuweisen. Im Beispiel oben ist z.B. “SLIDER_MIX” der Controller-ID 15 zugeteilt.

Wiederholen Sie diesen Schritt, bis Sie in Pure-Data allen gewünschten Tasten, Reglern und Knöpfen eine ID zugewiesen haben.

Schritt 4: Netzwerk-Einstellungen
Wenn PixelController auf dem **gleichen** Rechner wie PureData läuft, klicken Sie auf die Message-Box um eine Verbindung mit PixelController aufzubauen.

Läuft PixelController auf einem **anderen** Rechner als PureData, stellen Sie zunächst sicher, dass beide Rechner im selben Netzwerk einander erreichen, und tragen Sie die IP des Zielrechners in das weiße Textfeld ein (zuerst den Edit Modus aktivieren) und klicken Sie anschließend auf „<< PRESS HERE“

ACHTUNG!
An vielen MIDI-Geräten finden sich einige Regler, Taster und Knöpfe, die **keine MIDI-Daten** senden. Hierzu gehören meist der Main-Volume-Regler; z.B. „Up“ und „Down“-Tasten; an älteren Geräten auch schon mal der Pitch-Bend-Regler. Näheres dazu finden Sie in den Unterlagen zum Gerät).

Hinweis:
Detaillierte Erklärung, wie das Pure Data-Tool “MIDI2OSC” funktioniert finden Sie auf meinem Blog:
http://neophob.com/2012/12/use-a-midi-device-to-control-pixelcontroller/

Command Line Interface

Mit dem Command Line-Interface können alle Einstellungen von PixelController geändert werden.
In **Windows** startet man dazu `PixConCli.cmd` in einer Shell (cmd.exe),
unter **OSX** oder **Linux** kann das Shell-Skript `PixConCli.sh` verwendet werden.
Wenn Sie PixConCli ohne Parameter starten, wird eine Hilfe mit möglichen Parametern angezeigt.

Beispiel

Um alle Einstellungen zufällig auswählen zu lassen, ("Shuffler") müsste Ihr Code folgendermaßen aussehen:

```
# ./PixConCli.sh -c RANDOMIZE
PixelController Client v0.3
      127.0.0.1:3448 -> command: RANDOMIZE <>
Connected to 127.0.0.1:3448
Close connection, Bye!
```

Beispiel

Dabei kann der Aufruf auch Remote erfolgen, ein entfernter Rechner kann so ferngesteuert werden:

```
# ./PixConCli.sh -h 192.168.111.21 -p 3448 -c RANDOMIZE
PixelController Client v0.3
      192.168.111.21:3448 -> command: RANDOMIZE <>
Connected to 192.168.111.21:3448
Close connection, Bye!
```

Beispiel

Ein weiteres Beispiel, um Statistiken von PixelController auszulesen. Das kann bei Performance-Problemen hilfreich sein:

```
# ./PixConCli.sh -c JMX_STAT -p 1337
PixelController Client v0.3
      127.0.0.1:1337 -> command: JMX_STAT <>
Generic:
server version           : 1.1
current fps              : 19.992 (100% of configured fps: 20)
frame count              : 106
running since            : 0:00:05.474

The following average times have been collected during the last 10.004 seconds:
   generator             : 0.520ms
   effect                : 0.000ms
   output schedule       : 0.080ms
   fader                 : 0.000ms
   debug window          : 5.471ms
   output prepare wait   : 0.000ms
   output update wait    : 0.000ms
   matrix emulator window: 0.840ms

Ouput-specific average times for output #1: NULL (NullDevice)
   prepare               : 3.740ms
   update                : 0.000ms
```

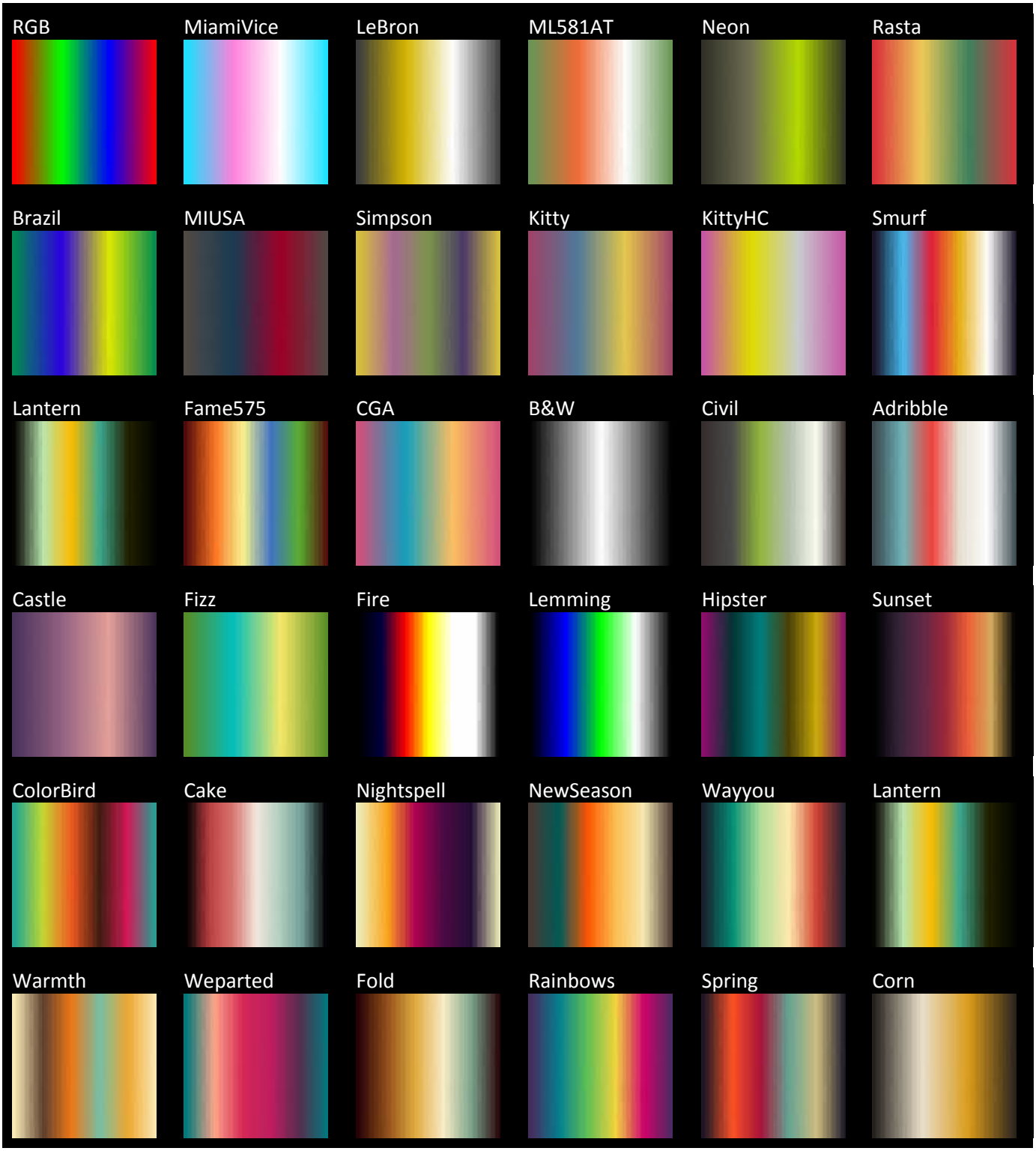
ACHTUNG!

Die Statistiken (-c JMX_STAT) müssen auf Port 1337 abgefragt werden!

Anhang A - Übersicht PixelController

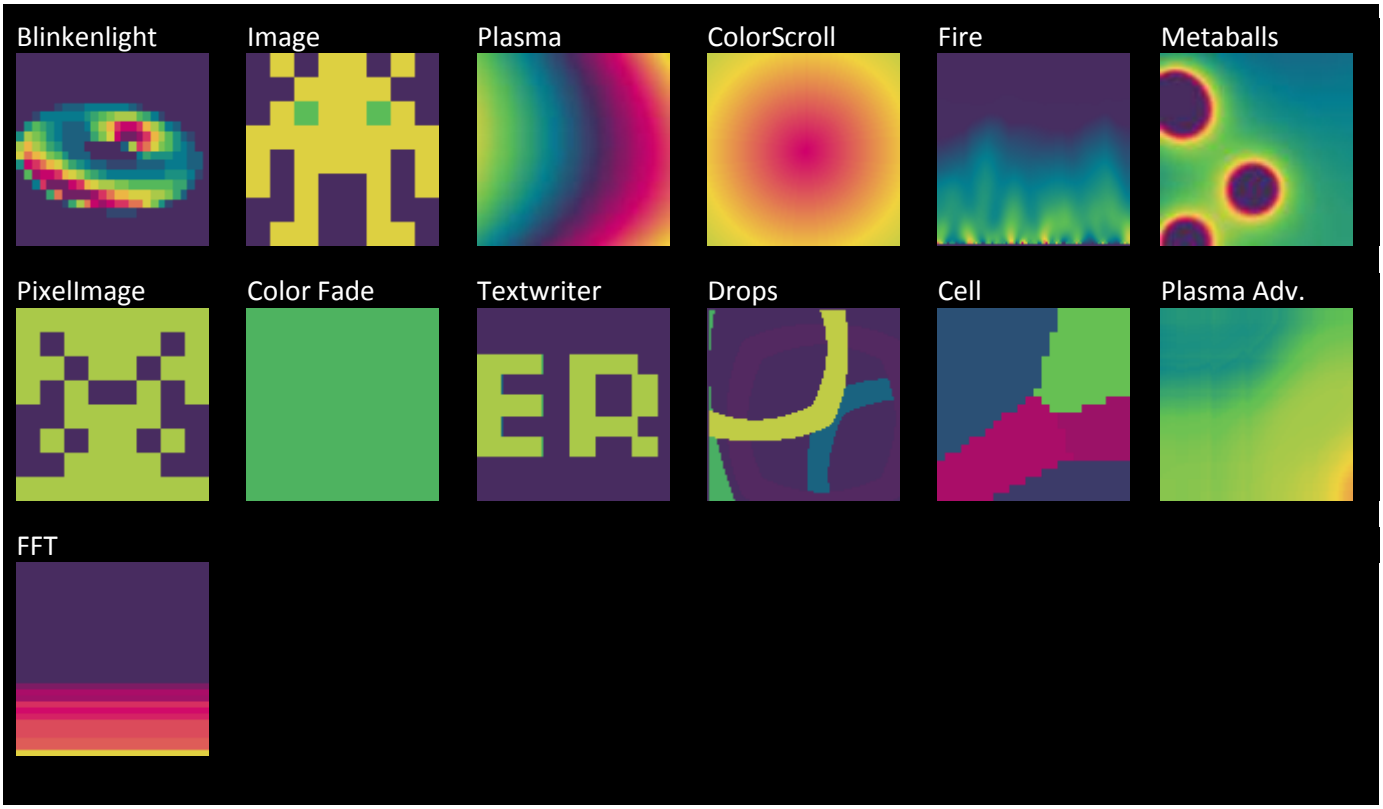
Übersicht Colorsets

Folgende Colorsets stehen Initial zur Verfügung:



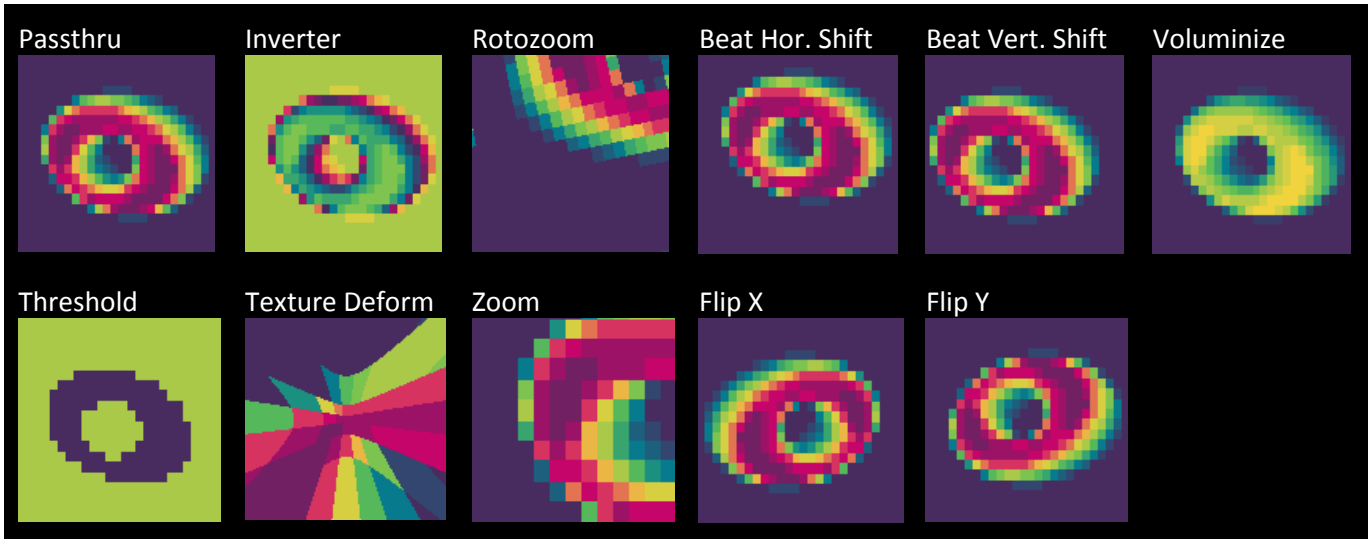
Übersicht Generatoren

Übersicht aller Generatoren mit dem “Rainbows” Colorset:



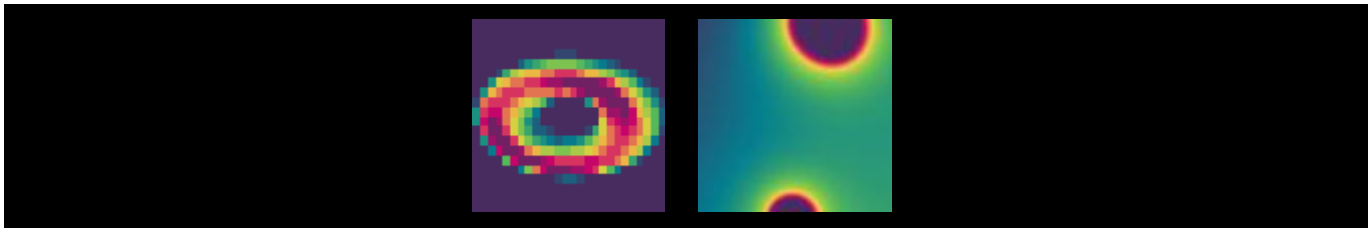
Übersicht Effekte

Übersicht aller Effekt mit dem Colorset “Rainbows”. Der Passthru Effekt modifiziert den Bildinhalt nicht - dieses Bild entspricht also dem Input Bild.

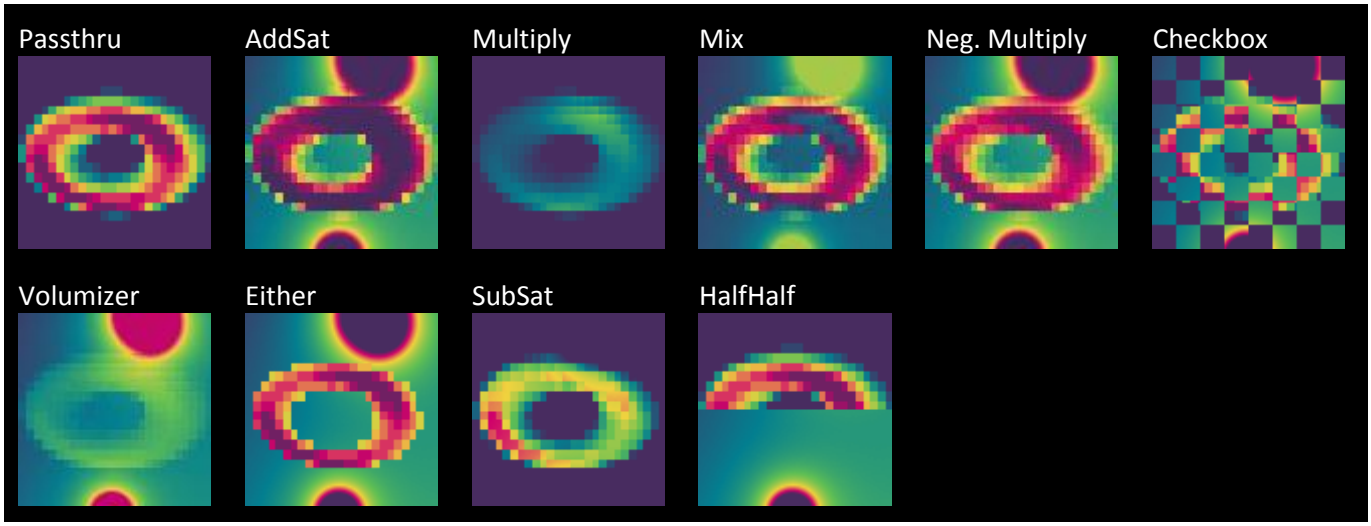


Übersicht Mixer

Aus folgenden zwei Generatoren



wurden folgende Outputs generiert



Anhang B - Arduino

Arduino-Setup

Wie in der Tabelle oben ersichtlich ist, liefert PixelController für diverse Hardware die Arduino-Firmware mit. Arduino ist ein quelloffenes Soft- und Hardwareprojekt, welches u.a. zur LED-Ansteuerung verwendet werden kann. Weiterführende Informationen sind auf http://arduino.cc/ ersichtlich. Verwendet und getestet wurde Arduino 1.0x.

Je nach verwendeter Firmware benötigt Arduino so genannte Libraries von anderen Herstellern/Entwicklern. Diese Libraries erweitern die Funktionalität von Arduino.
Auf http://arduino.cc/en/Guide/Libraries wird detailliert erklärt, wie man solche Libraries installiert.

Unter `data/ArduinoFW` sind alle Arduino-Firmware- und Library-Files abgelegt, die mit PixelController mitgeliefert werden.

Sie können mit einem Arduino-Board kostengünstig selber eine LED Matrix zusammenstellen. Alles was Sie dazu benötigen, sind “intelligente Pixel” (eine LED, welche sich die Farbe “merken” kann und direkt angesprochen werden kann). Der Autor und Hauptentwickler von PixelController bietet im PixelInvaders-Webshop solche Produkte an, unter anderem auch ein Do-It-Yourself-Kit mit einer detaillierten Anleitung, nach der das Panel zusammengebaut wird.

PixelInvaders Arduino-Firmware

Die PixelInvaders Firmware kann mit folgenden LED Modulen verwendet werden:

LED Modul	Firmware-Verzeichnis
LPD6803	`integration/ArduinoFw/pixelinvaders/neoLedLPD6803Spi`
WS2801	`integration/ArduinoFw/pixelinvaders/neoLedWS2801Spi`

Das einzige, was Sie in der Firmware einstellen müssen, ist die Anzahl der verwendeten Panels:

```
// ======= START OF USER CONFIGURATION =======
//define nr of Panels*2 here, 4 means 2 panels
#define NR_OF_PANELS 4
// ======= END OF USER CONFIGURATION =======
```

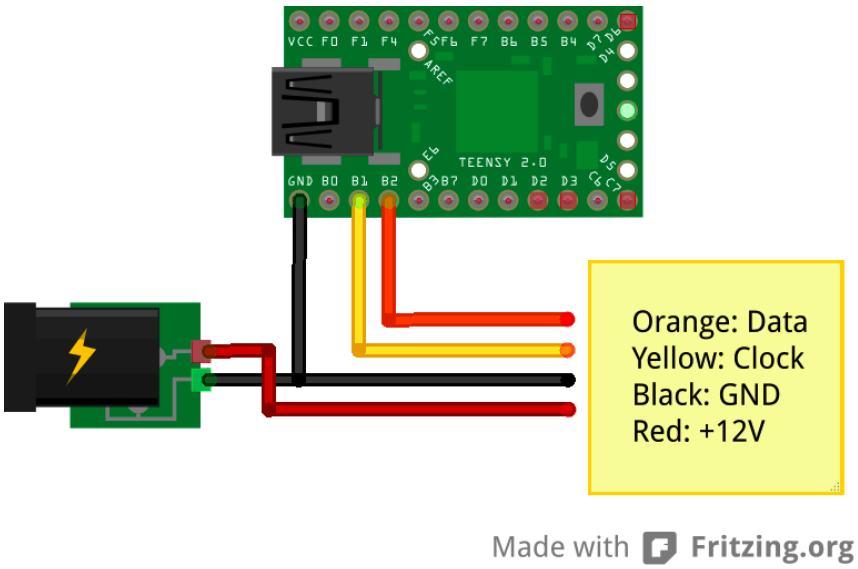
Die PixelController unterteilt eine LED-Matrix in zwei Bereiche auf, um die serielle Datenübertragung zu optimieren, daher muss die Definition der Panels (`NR_OF_PANELS`) entsprechend angepasst werden. Jedes serielle Paket enthält somit Daten für 32 Pixel.

✅ **Beispiel:**
Angabe für ein Panel: `#define NR_OF_PANELS 2`
Angabe für zwei Panels: `#define NR_OF_PANELS 4`
Angabe für acht Panels: `#define NR_OF_PANELS 16`
usw.

Teensy Boards

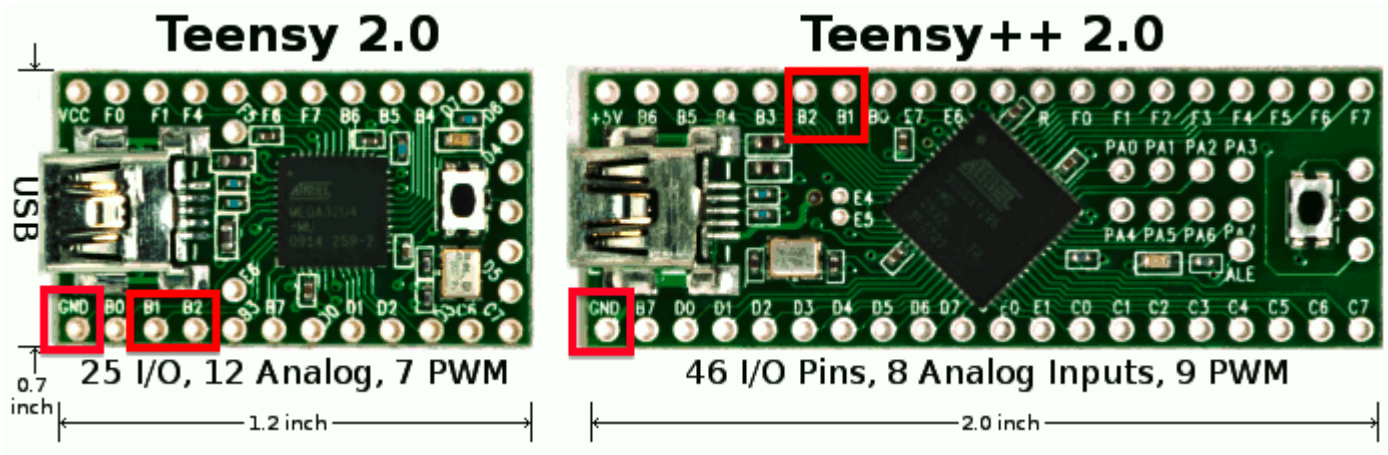
Ich empfehle, für die PixelInvaders Panels ein Teensy Board zu verwenden. Diese Arduino-kompatiblen Mikroprozessor Boards sind sehr klein und bieten im Vergleich zu original-Arduino-Boards einen besseren seriellen Durchsatz.

Folgendes Bild zeigt, wie das Teensy Board, die LED Module und das Netzteil verbunden werden.
Wichtig ist, dass alle 3 Komponenten die **gleiche Masse** (GND, - Pol) haben.



Hinweis:
Versuchen Sie, die SPI-**Datenkabel möglichst kurz** zu halten, um Probleme zu vermeiden. Siehe dazu folgenden Blog-Post: http://neophob.com/2012/04/long-distance-spi-installation-lessons-learned.

Die Hardware-SPI-Pins, welche mit den LED Modulen verbunden werden, sind im folgenden Bild sichtbar:



Serielle Latenz verschiedener Arduino-Boards

Das schwächste Glied bei der Verarbeitung der von PixelController gesendeten Daten ist Ihr Interface zur LED-Matrix. Hier geht es um Millisekunden. Dabei ist die Wahl des "richtigen" Arduino entscheidend.
Ich habe verschiedene Arduino-Boards verglichen, siehe http://neophob.com/2011/04/serial-latency-teensy-vs-arduino/ für Details.

Gemessen wurde die Zeit, die benötigt wird, um eine Nachricht vom Computer via USB auf einen Arduino und zurück zu senden. Bitte beachten Sie nur die **Native-Spalte**:

Bytes	Arduino UNO		Arduino 2009		Teensy 2.0	
	Native	Java	Native	Java	Native	Java
1	4.1	4.09	2	2.93	1	2.21
2	4.1	4.09	1.99	3.56	1	2.21
12	4.09	4.25	4	3.99	1	2.21
30	4.67	6.67	5.29	5.11	1	2.21
62	8.19	8.23	8	8.05	1.01	2.3
71	8.19	9.12	8	9.8	1.02	2.21
128	13.6	15.11	13.99	14.05	1.48	3.05
500	45.05	45.27	45.99	46.07	3	4.03
1000	87.72	89.96	88.99	89.92	5.81	6.73
2000	172.07	173.68	175.96	176.1	10	11.4
4000	344.04	344.03	348.8	349.77	19.99	21.07
8000	683.99	683.97	695.85	696.08	38.99	40

time value in ms

Der **Arduino 2009/Duemilanove** ist das älteste Arduino-Board im Test. Es verwendet den FTDI-Chip mit einem entsprechenden Treiber, nämlich den FTDI VCP (Virtual Com Port), zur Umwandlung der USB-Daten in serielle Daten. Weitere Informationen hierzu im nächsten Kapitel.

Der neuere **Arduino UNO** ist u.a. mit einem Atmega16U2-Prozessor ausgerüstet, welcher sich um die Datenumwandlung von USB zu seriell kümmert. Das hat den Vorteil, dass kein spezieller Treiber mehr benötigt wird, um das Board zu betreiben. Die Performance hat sich im Vergleich zum Vorgänger jedoch kaum verändert.

Das **Teensy 2.0-**Board ist eine Entwicklung von Paul Stoffregen. Dieses Board verwendet einen Atmel Chip, der die Umwandlung der USB/seriell-Daten direkt auf dem Mikroprozessor erledigt. Dies macht sich mit einer sehr kleinen Latenzzeit bemerkbar. Einer der neusten Arduinos, **Arduino Leonardo,** verwendet den gleichen Mikroprozessor wie Teensy 2.0. Ich nehme an, dass dieser eine ähnlich geringe Latenzzeit aufweist.

Mein **persönlicher Favorit** ist daher das **Teensy 2.0** Board. Dieses Board empfehle ich auch zur Verwendung mit einem oder mehreren PixelInvaders Panels DIY Panels.

Arduino Duemilanove / FTDI Treiber

Verwenden Sie einen älteren Arduino Duemilanove (mit FTDI-Treiber) sollten folgende Einstellungen geprüft werden[7] um die Latenz zu verringern:

OSX

Der OSX-FTDI-Treiber hat per Default einen Latenzwert von 2ms, eine Änderung des Treiberverhaltens ist nicht nötig.

> **Hinweis:**
> FTDI hat einen technischen Hinweis mit Informationen zum OSX-Treiber veröffentlicht: PDF.

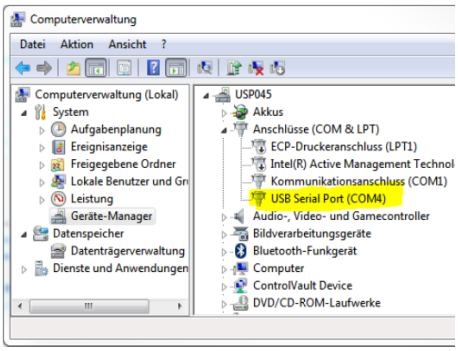
Linux

Die Latenz kann unter Linux via sys-Interface gesetzt werden:

```
# cat /sys/bus/usb-serial/devices/ttyUSB0/latency_timer 16
# echo 1 > /sys/bus/usb-serial/devices/ttyUSB0/latency_timer
# cat /sys/bus/usb-serial/devices/ttyUSB0/latency_timer 1
```

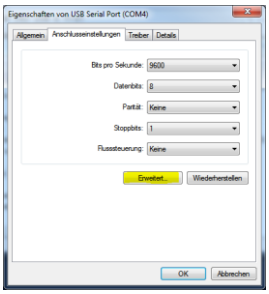
Windows

PixelController schließen, Arbeitsplatz->rechte Maustaste->Bearbeiten. Das Fenster Comuterverwaltung wird sich öffnen. Hier nehmen Sie die Latenzeinstellungen wie folgt vor:



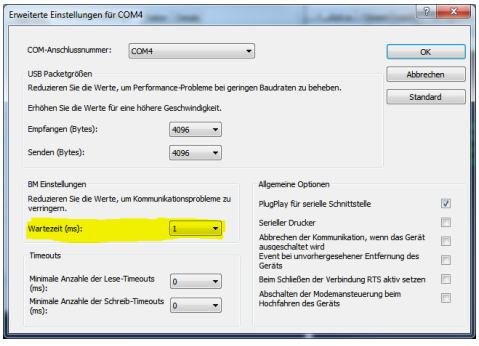
Den Gerätemanager starten und das serielle Interface auswählen.

Anschlüsse->USB-Serial Port



Auf den zweiten Reiter klicken. Sie sind dann bei den Anschlusseinstellungen.

Hier auf *Erweitert* klicken



In der linken, unteren Hälfte des Fensters finden Sie die Einträge zu den BM-Einstellungen. Hier die **Wartezeit** bei *Default* vom voreingestellten Wert, meist 16ms, auf 1ms heruntersetzen.

Anschließend alle Fenster mit OK wieder schließen und PixelController erneut starten. Durch die nun eingestellte, geringere Latenz sollten Ihre Panels jetzt wesentlich schneller reagieren und in Folge dessen auch weitgehend ruckelfrei laufen.

[7]Quelle: http://projectgus.com/2011/10/notes-on-ftdi-latency-with-arduino/

Glossar

Begriff	Details	Weiterführende Informationen
Applikation	Anwendungssoftware, kurz Anwendung; engl. application software, kurz App).	Webseite: wikipedia-Anwendungssoftware
Arduino	Eine kostengünstige Mikroprozessor-Platine mit passender Entwicklungssoftware, die beide quelloffen veröffentlicht wurden.	Webseite: arduino.cc
Betriebssystem	Das Betriebssystem ist ein, teilweise aus mehreren Programmen, bestehendes Computerprogramm, welches als Schnittstelle zwischen den einzelnen Hardware- Komponenten dient, und damit die Grundlage für den Rechner als arbeitsfähiges Werkzeug darstellt.	Webseite: wikipedia - Betriebssystem
FTDI **FTDI-Schnittstelle**	FTDI ist ein schottischer Entwickler und Hersteller von Halbleitern mit USB-Schnittstellen als Spezialgebiet.	Die FTDI-Schnittstelle konvertiert Datenströme der seriellen RS232-Schnittstelle in ein USB-Signal, was es den Nutzern erlaubt, andere, z.B. selbst gebaute, oder Geräte wie bspw. Arduino an einem PC oder Mac zu betreiben.
Gammakorrektur	Korrekturfunktion um lineare Farbverläufe dem menschlichen, nicht linear empfindlichen, Auge anzupassen.	Webseite: wikipedia - Gammakorrektur
GUI	Grafische Benutzeroberfläche. Sie dient dem Anwender zur Bedienung der Software.	
Java	Java ist eine objektorientierte Programmiersprache.	Webseite: wikipedia - Java-Programmiersprache
“Intelligente Pixel”	Als “Intelligente Pixel” werden LED-Pixel bezeichnet, die mit einem Logik-Chip ausgerüstet sind, der es Ihnen erlaubt, jedes Pixel einzeln anzusprechen.	
Interface	engl. für Schnittstelle.	Siehe “Schnittstelle”
Kommentar	Programmierer bedienen sich bei der Erstellung von Programmcode sog. Kommentare, die ihnen und dem Anwender beim Lesen des Quellcodes behilflich sein sollen. Je nach Programmiersprache unterscheiden sich die einen Kommentar einleitenden Zeichen.	✅ **Beispiele:** Arduino: // Kommentar bis Zeilenende Html: <!-- Kommentar -- > PHP: /* Kommentar */ Webseite: wikipedia - Kommentar_Programmierung
Midi, MIDI	Schnittstelle zur Übertragung bestimmter **Steuersignale** zwischen elektronischen	**M**usical **I**nstrument **D**igital **I**nterface

	Musikinstrumenten, Steuergeräten und/oder PCs mit Musiksoftware. Es wird **KEIN AUDIO** übertragen!	Webseite: wikipedia - MIDI
Output-Fenster	Der Teil der PixelController-Software, der Ihnen auf Ihrem PC-Bildschirm eine Vorschau dessen zeigt, was Sie mit Hilfe von PixelController auf Ihren Panels anzeigen lassen möchten.	
OSC	Die **O**pen **S**ound **C**ontrol - Schnitstelle zur für eine Netzwerk basierende Kommunikation, wird häufig in Musik- u. Multimedia Projekten verwendet.	Webseite: wikipedia - OSC hexler.net - touchosc
Plugin	Ein PlugIn ist ein Softwaremodul, das von einer Softwareanwendung während seiner Laufzeit entdeckt und eingebunden werden kann, um dessen Funktionalität zu erweitern.	Webseite: wikipedia - Plugin
Pure-Data Extended	Software zur objektorientierten, grafischen Programmierung von Audio-Umgebungen.	Webseite: puredata.info
PWM-Modulation	Mit Hilfe der **P**uls**W**eiten**M**odulation steuern wir die Helligkeit und Farben der LEDs. Die Pulsweitenmodulation *(PWM)* ist eine Modulationsart, bei der eine technische Größe (z. B. elektrischer Strom) zwischen zwei Werten wechselt. Dabei wird bei konstanter Frequenz der Tastgrad eines Rechteckpulses moduliert, also die Breite der ihn bildenden Impulse.	✅ **Beispiel:** **WS2801** ist eine 3-Kanal-Konstantstrom-Stromquelle mit programmierbaren PWM-Ausgängen Original-Bezeichnung: **WS2801** - 3-channel constant current LED Driver with programmable PWM outputs Webseite: wikipedia - Pulsweitenmodulation
Raspberry Pi	engl. Wortspiel für Himbeerkuchen Ein preiswerter Ein-Platinen-PC (easy PC).	Webseite: raspberrypi.org
Schnittstelle	Der Teil einer Software, der die Kommunikation zum z.B. Betriebssystem übernimmt.	Webseite: wikipedia - Schnittstelle
Source Code	Quellcode. Der für Menschen lesbare, in einer Programmiersprache geschriebene Text eines Computerprogramms.	Webseite: wikipedia - Quellcode
Teensy	Arduino-kompatible Mikroprozessor-Platine.	Webseite: prjc.com
Visual	Ein Visual ist eine in Echtzeit generierte Animation und ist beinhaltet folgende Komponenten: **Colorset**: Definierte Farb-Palette	

	Generator: Generiert den Bildinhalt **Effekt**: Modifiziert den Bildinhalt **Mixer**: Führt zwei Bildinhalte zusammen	
Zip-File	Eine komprimierte Datei. Zip-Dateien ermöglichen u.a. den Austausch von mehreren, meist größeren Dateien, die durch das Zip-Verfahren zusammengeführt und verkleinert werden, um den Transport zu beschleunigen.	Andere Formate sind z.B. rar, taz, tar, u.a. Der PixelController-Download wird als Zip-Datei angeboten. Webseite: wikipedia - ZIP-Dateiformat

Übersicht der im Handbuch verwendeten Links

PixelController
Webseite: http://pixelinvaders.ch
Kontakt: michu@neophob.com

Software-Download
http://code.google.com/p/pixelcontroller/downloads/list

Seeedstudios Rainbowduino V3
http://seeedstudio.com/wiki/Rainbowduino_v3.0

ArtNet Devices
http://artisticlicence.com/

MiniDmx Devices
http://led-studien.de

Generic UDP Devices
https://github.com/scottjgibson/PixelPi

TPM2 Serial devices
http://ledstyles.de
http://ledstyles.de/fpost296621.html

TPM2 Net devices
http://ledstyles.de/fpost296621.html

Element Labs Stealth LED panel
http://cled.barcousa.com/support/STEALTH/STEALTH_Users_Guide.pdf

Zip-Software
http://winzip.de
http://winrar.de/

librxtx
https://github.com/neophob/librxtx/archive/v2.2.tar.gz

Applikation unter Linux selber zu kompilieren http://de.wikihow.com/Wie-man-in-Linux-ein-Programm-kompiliert.

PixelController-Basis-Pack kaufen
http://shop.pixelinvaders.ch/product/pixelinvaders-diy-basic-pack

Neorainbowduino Firmware
http://code.google.com/p/neorainbowduino/

Rainbowduino-v3-streaming-Firmware

https://code.google.com/p/rainbowduino-v3-streaming-firmware/

PixelPi Software
https://github.com/scottjgibson/PixelPi

Farb-Palette Designer
http://colorschemedesigner.com
http://palette.ws/
https://kuler.adobe.com/

Blinkenlights
http://blinkenlights.net/

Processing
http://processing.org/

Blinkenlights Processing Plugin
http://robinsenior.com/blinkenlights/

Blinkenlight Movie Converter
http://www.capybara.org/~dfraser/archives/261

LineIn
http://www.rogueamoeba.com/freebies/

Windows default input-device
http://blogs.creighton.edu/bluecast/tips-and-tricks/set-the-default-microphone-and-adjust-the-input-volume-in-windows-7/

Input-Device für ALSA
http://dsl.org/cookbook/cookbook_27.html

Audio Input per Line-In
OSX: http://rogueamoeba.com/freebies/

Fehlersuche
https://github.com/neophob/PixelController/issues

OSC-Schnitstelle
http://en.wikipedia.org/wiki/Open_Sound_Control

TouchOSC
http://hexler.net/software/touchosc

PureData-Extended
http://puredata.info/

PixelController und MIDI

http://en.flossmanuals.net/pure-data/

PureData-Tool “MIDI2OSC”
http://neophob.com/2012/12/use-a-midi-device-to-control-pixelcontroller/

Arduino
http://arduino.cc/

Arduino-Firmware
http://arduino.cc/en/Guide/Libraries

PixelInvaders Webshop
http://shop.pixelinvaders.ch

Latenz bei Arduino
http://neophob.com/2011/04/serial-latency-teensy-vs-arduino/

Teensy 2.0-Board
http://www.pjrc.com/teensy/index.html

FTDI und OSX
http://www.ftdichip.com/Support/Documents/TechnicalNotes...0FTDI-Driver.pdf

Arduino Duemilanove / FTDI-Treiber
http://projectgus.com/2011/10/notes-on-ftdi-latency-with-arduino/