PixelController

PixelController - a matrix control project by Michael Vogt , (c) 2010-2013. The main goal of this application is to create an easy to use matrix controller software which creates stunning visuals!

Primary Website: http://www.pixelinvaders.ch

My Blog: http://www.neophob.com

Facebook: https://www.facebook.com/PixelInvaders

You can **download** PixelController on Google Code: http://code.google.com/p/pixelcontroller/downloads/

HOWTO USE PIXELCONTROLLER

Prerequisite:

- Java Runtime, v1.6+

You can start PixelController with an integrated GUI by double click on `PixelController.jar` or you can start the console version (for example on a Raspberry PI) of PixelController by executing the `console\PixelController.sh` (OSX/Linux) or `console\PixelController.cmd` (Windows) Script.

By default PixelController has **no configured** output device (= no configured LED Matrix). To change that open the `data/config.properties` configuration file and make the necessary changes, lines starting with # are ignored. The most important parts are:

```
output.resolution.x=8
output.resolution.y=8
```

which defines the resolution of your matrix. And you need to define an Output device, for example for the PixelInvaders panels:

```
pixelinvaders.layout.row1=NO_ROTATE,ROTATE_180
#pixelinvaders.layout.row2=NO_ROTATE,NO_ROTATE
```

this defines two PixelInvaders panels while the second panel is rotated by 180 degrees. Take a look at the config file, there are alot of hints how to configure PixelController.

Main idea

A Visual can be assigned to one or more Output LED Matrices. A Visual consists of two **Generators** (create the content), two **Effects** (modify the content), a **Mixer** (mix the content) and a **Colorset** (define the look of the content). I try to visualize it:

```
[GENERATOR A] ---> [EFFECT A] ---> [MIXER] <--- [EFFECT B] <--- [GENERATOR B]
                                       |
                                       V  [Colorset]
                                   [VISUAL]
```

Per default PixelController creates one Visual more than the number of connected Output devices. This allows you to play with a non-visible Visual, that can be displayed later. All Visuals can be stored (and of course loaded) in a preset.

DEMO

Check out PixelController Rough Cut #2. Featuring two PixelInvaders panels, PixelInvaders 3D RGB Panels and PixelInvaders panels controlled by a tablet (OSC) to see PixelController in action on two PixelInvaders panels.

SUPPORTED HARDWARE

PixelController supports different (LED) matrix hardware devices/controller:

- PixelInvaders 3D Panels serial device (see Readme.PixelInvaders, http://www.pixelinvaders.ch)
- PixelInvaders 3D Panels network device (see Readme.PixelInvaders, http://www.pixelinvaders.ch)
- Seeedstudios Rainbowduino V2 (see Readme.rainbowduinoV2)
- Seeedstudios Rainbowduino V3 (Using this firmware: https://code.google.com/p/rainbowduino-v3-streaming-firmware)
- ArtNet Devices, multiple universe are supported,510 Channels (170 RGB Pixels) per universe
- MiniDmx Devices (like the SEDU board of http://www.led-studien.de)
- Element Labs Stealth LED panel. No longer in production ()
- Generic UDP Devices (for example Raspberry Pi, check out the PixelPi Software)
- TPM2 Serial devices (see http://www.led-studien.de for more information)
- TPM2 Net devices (see http://www.led-studien.de for more information)
- E1.31 devices (see http://www.opendmx.net/index.php/E1.31)

Check out the `integration/ArduinoFW` directory, all Arduino based firmware files are stored there.

Which firmware should I use?

If you don't have a hardware controller (like ArtNet or E1.31) and would like to use an Arduino/Teensy microcontroller you can choose between different firmwares.
* If you bought a PixelInvaders DIY Kit, use the `integration/ArduinoFw/pixelinvaders/neoLedLPD6803Spi` firmware * If you want to create a ONE panel matrix with an arbitrary resolution, use the `integration/ArduinoFw/tpm2serial` firmware * If you want to create multiple 8x8 panels, use the `integration/ArduinoFw/pixelinvaders/neoLedWS2801Spi` firmware

I recommend a Teensy 2.0 microcontroller, as some Arduino boards suffer from bad serial latency (especially the Arduino UNO r3). You need to install the Arduino IDE, see the "Getting started with Arduino" (http://arduino.cc/en/Guide/HomePage) Tutorial.

You need to know how to install an Arduino Library (http://arduino.cc/en/Guide/Libraries). For PixelInvaders Panels (LPD6803) install the `integration/ArduinoFw/libraries/timer1` and `integration/ArduinoFw/libraries/neophob_lpd6803spi` libraries, for other panels (WS2801, WS281x...) install the `integration/ArduinoFw/libraries/FastSPI_LED2` library.

How does it work?

PixelController generates the content for the LED matrix, sends the data out to the controller, the controller will update the LED modules. There are two options for "sends the data": * sends the data via USB to the Arduino/Teensy board aka. DIY LED controller. * sends the data via ethernet to a PixelInvaders/E1.31/ArtNet... device.

Here are some primitive schemes:

```
[PixelController]---<USB>---[Teensy with PixelInvaders firmware]---<SPI>---[LED#1]---[LED#2]...

[PixelController]---<USB>---[Teensy with TPM2 firmware using fastspi2 lib]---<SPI>---[LED#1]---[LED#2]
...

[PixelController]---<ethernet>---[Artnet Controller]---<???>---[LED#1]---[LED#2]...
```

Advanced PixelController configuration

There are a lot of options in the `config.properties` file. I describe some examples, PixelController updates all Visuals depending on the Sound input. If a beat is detected, the Visuals are updated faster. You can disable this behaviour by setting this option:

```
#=========================
#enable pixelcontroller sound analyzer (disable it if you don't have a sound card)
#=========================
sound.analyze.enabled=true
```

There is a Generator called "Screen Caputure" which is disabled by default. If you want to enable this generator, edit the following settings:

```
#x/y offset for screen capturing generator
#if you define screen.capture.window.size.x as 0, the screen capture generator will be disabled
screen.capture.offset=100
screen.capture.window.size.x=500
screen.capture.window.size.y=300
```

This enables the Screen Caputure Generator which captures a region of 500 x 300 pixels. Potential use cases for this Generator are: YouTube videos, other movie players...

Or you can start PixelController in the random mode where PixelController changes the Visuals randomly:

```
#=========================
#start in random mode?
#=========================
startup.in.randommode=false
```

Or you can save a preset and load that one per default if you start PixelController (per default, preset 0 will be loaded)

```
#=========================
#load a preset if PixelController starts?
#Warning, this will overwrite your settings configured above (initial generator values)!
#=========================
#startup.load.preset.nr=1
```

You can define the size of the PixelController GUI, for example the size of the simulated LED Matrix (which is per default 16 pixels):

```
#=========================
#the size of the software output matrix
#=========================
led.pixel.size=16
```

Or define the window size, depending on this setting, the Visuals are displayed larger or smaller.

```
#=========================
#define the maximal window size (control window)
#=========================
gui.window.maximal.width=820
gui.window.maximal.height=600
```

You can define your own Colorsets, they are defined in the file `data/palette.properties` . A Colorset definition consists of a name and multiple RGB color values. Here is an example:

```
MiamiVice=0x1be3ff, 0xff82dc, 0xffffff
```

There are more options in the config file, take a look - each option should be documented.

FRONTENDS

There are different frontends for PixelController (besides the GUI frontend):

- PixConCli: Command Line Interface for PixelController, works also remote. The CLI tool is called `PixConCli.cmd` on Windows and `PixConCli.sh` on Linux/OSX.
- OSC: The OSC interface of PixelController is listening (by default) on port 9876. Processing examples are included how to communicate with PixelController via OSC protocol. Or create your own interfaces, for example with the great TouchOSC application or using PureData or MaxDSP.

CLI EXAMPLES

You can send OSC messages to PixelController to control the software. PixelController includes a simple CLI tool to control the software by console. Start PixelController, then open the console:

Randomize current Visual

```
    # ./PixConCli.sh -c RANDOMIZE
```

Select Image Generator as Generator A (0 is Passthru, 1 is Blinkenlights...) for current Visual:

```
    # ./PixConCli.sh -c CHANGE_GENERATOR_A 2
```

Load image gradient.jpg

```
    # ./PixConCli.sh -c IMAGE gradient.jpg
```

OSC MESSAGES

Here are all commands PixelController knows.

```
    CHANGE_GENERATOR_A          # of parameters: 1     <INT> change first generator for current visual
    CHANGE_GENERATOR_B          # of parameters: 1     <INT> change first generator for current visual
    CHANGE_EFFECT_A             # of parameters: 1     <INT> change first effect for current visual
    CHANGE_EFFECT_B             # of parameters: 1     <INT> change second effect for current visual
    CHANGE_MIXER                # of parameters: 1     <INT> change mixer for current visual
    CURRENT_VISUAL              # of parameters: 1     <INT> select actual visual
    CURRENT_COLORSET            # of parameters: 1     <INT> select actual ColorSet
    GENERATOR_SPEED             # of parameters: 1     <INT> generator speed 0 .. 200 (default speed is 100)

    CHANGE_OUTPUT_VISUAL        # of parameters: 1     <INT> change visual for current output
    CHANGE_OUTPUT_FADER         # of parameters: 1     <INT> change fader for current output
    CHANGE_ALL_OUTPUT_VISUAL    # of parameters: 1     <INT> change visual for all outputs
    CHANGE_ALL_OUTPUT_FADER     # of parameters: 1     <INT> change fader for all outputs
    CURRENT_OUTPUT              # of parameters: 1     <INT> select current output

    BLINKEN                     # of parameters: 1     <STRING> file to load for the blinkenlights generator
    IMAGE                       # of parameters: 1     <STRING> image to load for the simple image generator
    TEXTDEF                     # of parameters: 1     <INT> select texture deformation option, 1-11
    ZOOMOPT                     # of parameters: 1     <INT> select zoom options 1-4
    COLOR_SCROLL_OPT            # of parameters: 1     <INT> select color scroll fading direction, 1-14
    TEXTWR                      # of parameters: 1     <STRING> update text for textwriter generator
    TEXTWR_OPTION               # of parameters: 1     <INT> set mode textwriter (pingpong scroller, left scroller)
    CHANGE_BRIGHTNESS           # of parameters: 1     <INT> output brightness 0 .. 100
    GENERATOR_SPEED             # of parameters: 1     <INT> generator speed 0 .. 200 (default speed is 100)
    BEAT_WORKMODE               # of parameters: 1     <INT> change beat workmode 0-2
    OSC_GENERATOR1              # of parameters: 1     <BLOB> contains 4096 bytes (64x64x8bpp) or 12288 bytes (64x64x24bpp) of image data (depending on internal size)
    OSC_GENERATOR2              # of parameters: 1     <BLOB> contains 4096 bytes (64x64x8bpp) or 12288 bytes (64x64x24bpp) of image data (depending on internal size)

    CHANGE_THRESHOLD_VALUE      # of parameters: 1     <INT> select current threshold for the threshold effect, 0-255
    CHANGE_ROTOZOOM             # of parameters: 1     <INT> select angle for the rotozoom effect, -127-127

    CHANGE_PRESENT              # of parameters: 1     <INT> select current present id
    CHANGE_SHUFFLER_SELECT      # of parameters: 18    <INT>, parameter contains 15 nibbles to enable or disable the shuffler option (gets changed in the random mode), 0=OFF, 1=ON, example: 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1
    SAVE_PRESENT                # of parameters: 0     <NO PARAM> save current present settings
    LOAD_PRESENT                # of parameters: 0     <NO PARAM> load current present settings
    RANDOM                      # of parameters: 1     <ON|OFF> enable/disable random mode
    RANDOM_PRESET_MODE          # of parameters: 1     <ON|OFF> enable/disable random preset mode
    RANDOMIZE                   # of parameters: 0     <NO PARAM> one shot randomizer
    PRESET_RANDOM               # of parameters: 0     <NO PARAM> one shot randomizer, use a pre-stored present
    JMX_STAT                    # of parameters: 0     <NO PARAM> show JMX runtime statistic, default port: 1337 (use the -p switch)
    SCREENSHOT                  # of parameters: 0     <NO PARAM> save screenhot
    FREEZE                      # of parameters: 0     <NO PARAM> toggle pause mode
    TOGGLE_INTERNAL_VISUAL      # of parameters: 0     <NO PARAM> show/hide internal visual to save CPU
```

IT DOES NOT WORK!

Try to understand **WHAT** does not work, which component? is it the frontend? PixelController itself? or no output?

Here are some common errors:

- Is Java installed on your system? Open a terminal Windows (cmd.exe on Windows, terminal on OSX) and enter "java -version".
- Did you forgot to **edit the configuration file** `config.properties` . Take a look at the config examples files in the

`data/config.examples` directory!
- Did you flash the **correct firmware** to your Arduino/Teensy?
- **PixelInvaders panels**: Make sure that the Panel shows an **animated rainbow pattern** when the panels are powered on (make sure that you also power the Arduino/Teensy board). If you don't see a animated rainbow, make sure the directon of the modules is correct and that the Arduino/Teensy, LED modules and PSU share common ground. Verify the Arduino IDE don't spit out errors when you upload the firmware to the teensy
- **PixelInvaders panels**: Multiple users reported that the PixelInvader firmware did not work on a new Arduino UNO r3 board. I think the reason for this is the big serial latency. However using a Arduino UNO r1 worked flawlessly. Technically this is not a big deal, as the timeout value cold be adjusted in the firmware. Use a Teensy 2 board for best results.
- Make sure you're using an up-to date Java Runtime (JRE), this usually helps if the JVM crashes.
- If you use an extra long USB Cable (more than 5 meter) you might discover strange issues, try to use a short cable especially if you're uploading a firmware to the Arduino/Teensy.
- The **OSC Generator** does not work: make sure you select the correct resolution for the OSC sender, take a look at the INFO tab, there you see the PixelController internal buffer size. Use this resolution in your OSC sender (or Processing sketch).

HOWTO BUILD PIXELCONTROLLER

Prerequisite:

- Maven v2.x (if you use Maven 3, make sure to read http://neophob.com/2011/11/maven-3-is-evil/ first!)
- JDK 1.6+

Then run

```
# mvn initialize
to install the needed packages in your local repo and
# mvn clean package
to build PixelController, the distribution directory is "target/assembly/PixelController-VERISON/".
```

Hint: if you're using eclipse and you see an error like this `java.lang.NoClassDefFoundError: Could not initialize class gnu.io.RXTXVersionjava.lang.NoClassDefFound` make sure you add the lib/serial directory as "Native library location"

ADD NEW HARDWARE SUPPORT

It should be pretty simple to add support for new hardware. All Output code should go into the com.neophob.sematrix.output package (`src/main/java/com/neophob/sematrix/output` directory). All you need to do in the Output class is, take an array of int's (one int is used to store the 24 bpp) and send this buffer to your output device (via serial port, ethernet, bluetooth...). Maybe you need to reduce the color depth, flip each second scanline due hardware wiring, such helper methods should go into the `OutputHelper.java` class.

As a string point, add your hardware in the `OutputDeviceEnum.java` class and have a look where the other entries are referenced. **Take a look at the existing Output classes**, this should help you!

NEW RELEASE

Update Changelog, add git status:

```
# git diff v1.5.0 develop --stat
```

Update `readme.pdf` - use `README.md` as source.

Optional, license header check for all source files (http://code.google.com/p/maven-license-plugin/wiki/HowTo)

```
# mvn license:check -Dyear=2013 -Demail=michu@neophob.com (check)
# mvn license:format -Dyear=2013 -Demail=michu@neophob.com (apply)
```

Use the Maven version plugin to update your POM’s versions:

```
# mvn versions:set -DnewVersion=1.5.1
```

Rebuild:

```
# mvn clean deploy
```

Test application, make sure the `config.properties` file is correct.

Commit and push new version:

```
# git commit pom.xml -m "release v1.5.1"
# git push
```

Tag the release branch:

```
# git tag -a v1.5.1
# git push --tags
```

Merge into the master branch and push:

```
# git checkout master
# git merge develop
# git push
```

Checkout the master branch (already done)

Do a deployment build:

```
# mvn clean deploy
```

Release

PERFORMANCE

With the JMX interface you can monitor the status of your PixelController instance in real time. This will provide you with useful data such as required time for each layer (generator, effect, mixer…), the frame rate of your instance, allowing you to diagnose problems or performance issues. To read the JMX data, you will need to use a JMX client or the PixConCli util.

Example how to use PixConCli:

```
localhost:PixelController-1.3-SNAPSHOT michu$ ./PixConCli.sh -c JMX_STAT -p 1337
Create an RMI connector client and connect it to the RMI connector server 127.0.0.1:1337
Get an MBeanServerConnection...

Generic:
server version           : 1.1
current fps              : 20,036 (100% of configured fps: 20)
frame count              : 1771
running since            : 0:01:28.980

The following average times have been collected during the last 10.007 seconds:
   generator             : 0,310ms
   effect                : 0,000ms
   output schedule       : 0,140ms
   fader                 : 0,000ms
   debug window          : 15,210ms
   output prepare wait   : 0,005ms
   output update wait    : 0,005ms
   matrix emulator window: 0,440ms

Ouput-specific average times for output #1: NULL (NullDevice)
   prepare               : 1,550ms
   update                : 0,000ms

Close the connection to the server
```

CREDITS

- **Michael Vogt**: Project Lead, Main Developer
- **Markus Lang**: Maven enhancements, Output enhancements, Performance enhancements, Rainbowduino V3 support
- **McGyver666**: Contributor
- **Rainer Ostendorf**: Artnet Output
- **Pesi**: miniDMX Output, Tester
- **Scott Wilson**: Arduino/Rainbowduino Howto
- **Noxx6**: Bugfixes
- **okyeron**: Stealth output device
- **Dr. Stahl**: Documentation, Tester