Henry's Bench

JADAOL ...
$11.95

VICTSING ...
$10.79

OMORC ...
$6.99

Sections

Select Language Powered by Google Translate

U8glib Arduino OLED Tutorial 2: Playing with the Picture Loop

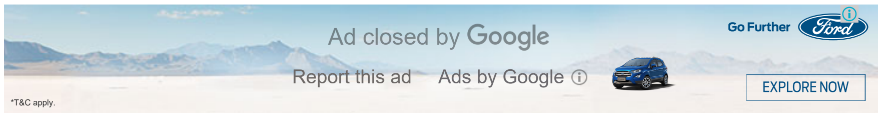
Contents [hide]

Why the Picture Loop?

I'm not the author of the library, so I don't dare wade into the minutia, but it's fairly easy to understand if you consider the challenge that he took up. Specifically, he chose to create a graphics library to support a wide variety of devices while maintaining a uniform set of graphics commands.

As he states in his wiki, some devices will allow you to turn a single pixel on. Others don't. Instead only send eight bits at a time.

Might there be some necessary compromises with a one size fits all approach? I'll bet there are.



Useful? For me, it absolutely is. I can pick a variety of displays for the various things that I build without relearning yet another graphics library.

You see, I like to spend energy on my projects measuring things, moving things, and controlling things. I want my fan to come on when things get hot and a heater to come on when they are cool. To do make that happen, I'm willing to toil over the temperature control algorithm.

What I'm not signed up for is digging into a display driver data sheet just so that I can provide a display. What I am willing to do is learn an understandable interface to provide an attractive project display that says that it's 44 degrees C and that my fans are on.

To that end, the U8glib works and in my humble opinion, is worth the minor effort that is necessary to master it.

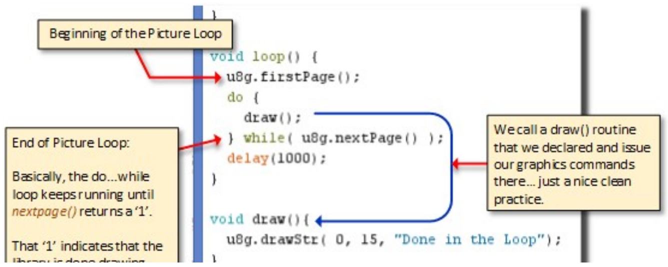
Therein lies the emphasis on the Picture Loop. In order to get productive use of the Picture Loop, you need to get a sense of how it works.

firstPage() and nextPage(): The Picture Loop Book Ends

A call to firstPage() marks the beginning of your picture loop.

Following this call, you're going to create a 'do…while' loop. Within this loop, you're going to issue draw commands. These drawing commands **must** be inside this loop. While the binary gods may occasionally smile on you if you violate this rule, I can promise that your results (if you get any) will be VERY unpredictable.

The programmer designed his library to work properly IF drawing commands occur inside the Picture Loop.



```
}
void loop() {
  u8g.firstPage();
  do {
    draw();
  } while( u8g.nextPage() );
  delay(1000);
}

void draw(){
  u8g.drawStr( 0, 15, "Done in the Loop");
}
```

Beginning of the Picture Loop

End of Picture Loop:

Basically, the do…while loop keeps running until *nextpage()* returns a '1'.

That '1' indicates that the

We call a draw() routine that we declared and issue our graphics commands there… just a nice clean practice.

JADAOL CAT 7 ETHERNET ...
(733)
$11.95
Shop now

VICTSING
$10.79
OMORC
$6.99

Jadaol Cat 7 Ethernet Cable 25 ft White -Shielded ...
$11.95

library is done drawing what you asked it to draw.
At that point you're free to go read some sensor input

Put Everything You Want To See In the Loop

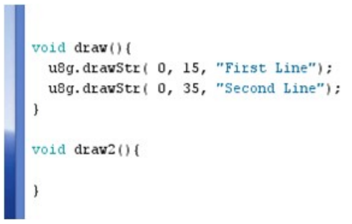
Each iteration of the picture loop creates a new screen. For example you cannot draw a box in the first picture loop and then a circles in another and expect them to show up on the same display. They won't. Only the circle will show up. In fact, each picture loop erases the output of the previously picture loop entirely.

Go ahead a paste the following sketch into your Arduino. Be sure to edit for your constructor.

```
#include "U8glib.h"

//****************************************************
// Change this constructor to match your display!!!
U8GLIB_SH1106_128X64 u8g(4, 5, 6, 7);
//****************************************************

void setup() {
  u8g.setFont(u8g_font_unifont);
  u8g.setColorIndex(1); // Instructs the display to draw with a pixel on.
}

void loop() {
  u8g.firstPage();
  do {
    draw();
  } while( u8g.nextPage() );
  delay(1000);
  u8g.firstPage();
  do {
    draw2();
  } while( u8g.nextPage() );
 delay(1000);
}


void draw(){
  u8g.drawStr( 0, 15, "First Line");
}

void draw2(){
  u8g.drawStr( 0, 35, "Second Line");
}
```

Upload the sketch and view the results. You'll notice that the 'First Line' and the 'Second Line' NEVER appeared on the display at the same time. In other words, the Picture Loop is functionally a command to paint a fresh display on a fresh canvas.

Now, remove the statement *u8g.drawStr(0, 35, "Second Line")* from **draw2()** and put it into **draw()** as shown below.



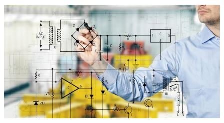
```
void draw(){
  u8g.drawStr( 0, 15, "First Line");
  u8g.drawStr( 0, 35, "Second Line");
}

void draw2(){

}
```

Upload the sketch and view the results.

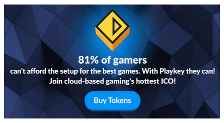
If you were successful, both 'First Line' and 'Second Line' will appear on the display at the same time for about one second. The display of both lines will immediately be followed by blank screen for one second.

Shop Related Products

In other words, an empty Picture Loop is the functional equivalent of a 'Clear Screen'.

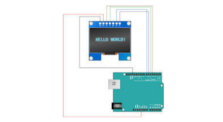

Power Supply Design Basic
Ad micro.rohm.com

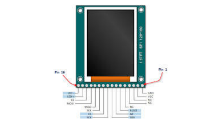
Arduino Displays
capnfatz.com


81% of gamers
can't afford the setup for the best games. With Playkey they can! Join cloud-based gaming's hottest ICO!
Buy Tokens
Playkey ICO Token Sale
Ad Playkey

U8glib Arduino OLED Tutorial 1: Hello World on Steroids
capnfatz.com

Race Up the Career Ladder
Ad Jigsaw Academy


OLED Arduino Blue 1.3" SH1106 Display – Quick Start Tutorial
capnfatz.com


1.8TFT V1.0 ST7736 Arduino Tutorial 1: Hello World
capnfatz.com

Arduino Adafruit GFX Library User Guide
capnfatz.com

Share this:

G+ Share 0 Save 2 submit Tweet More

7 COMMENTS

August 30, 2017 *Brandon*

this guys tutorials have helped me several times "Great stuff ..thank you"

January 22, 2017 *Mihai Tintea*

Most fun is when you insert a delay() statement in the do { …. } while (u8g.nextPage()) loop, to see how the pages are drawn slowly.

Aven 17010 Adjustable Circu…
$11.34
(815)

NEXTPAGE Brother Compatible Lamina…
$14.99
(71)

Resistance Loop Band Manual: Total Bod…
$2.99
(3)

Elegoo 37-in-1 Sensor Module Kit for A…
$29.86
(333)

Ads by Amazon

January 13, 2017 *LDP*

Delving just a tad deeper into the reasoning behind these display pages, they are provided to reduce memory (RAM) footprint of the display use.
Since the display cannot be written into directly or easily, one needs to have the display content (pixels) cached in memory. However, it is not always possible to cache the entire display content in memory due to the size of the display. For example, the common 128×64 wonder has 8K pixels whose values can be stored in 1KB of memory. If the microprocessor has access to let's say only 2KB of RAM, the display will rob more than half of that.
Hence the display cache paging idea. The display cache is made smaller (the default for the 128×64 display is 8 pages with 128 bytes each) to avoid using up too much of the already limited amount of RAM.
Every time the call to nextPage() is made, the cache is transferred (via selected protocol, such as I2C or SPI) to the display along with the page ID and that part of the display is drawn. So for example, for the default 128×64 display, one needs to call nextPage() 8 times to refresh all 8 sections of the display.
And since the cached page in memory can only hold pixels for one page at a time, it must be refilled with data for each consecutive nextPage() call. The drawing routines in the display library make sure that when they are called to draw something on the display, they project the pixels only to the active page that will be transferred to the display next. Which necessitates multiple executions of the drawing routine. The pixels that are part of the page will be stored in the display page cache and the others will be simply ignored. It feels kind of like drawing through a mask/template. Pretty much like spray painting signs with a template.
So while this library allows for tremendous memory savings, it does not come without a cost. The memory is saved at the expense of code size and of course execution speed.
An advanced programmer should be able to take advantage of this display pagination and modify the code (and the library code) to improve the speed of drawing.
Cheers.

December 23, 2016 *Herni*

Gracias, estoy tratando de entender los comandos de esta libreria

December 5, 2016 *César Fois*

Muito obrigado meu amigo, eu gostei sua explicação.

Thanks a lot my friend, i liked your explanation.

César

June 15, 2016 *shamika dalvi*

It worked great for me when i was trying to display images, thankyou for this code

January 22, 2016 *Ric*

Brilliant – I was struggling with the concept of this but your simplified description of the loop has opened up a whole 'new' world of hurt!!! – Namely porting some code that uses the Adafruit1306 library and leaving behind its hungry buffer gobbling ways.

ADD A COMMENT

Your email address will not be published. Required fields are marked *

Message*

Name*