BaX-lib

- The manual -

version 1

BaX-lib version 1.1.1 (April-2011)

Made by Binyamin Chai

Published by “ibnatfatil”

(this page is was left blank intentionally)

Contents

Introduction

The BaX-library (or BaX-lib for short) is a fast and simple 24bit bitmap library for C++ (g++ 4.x). Its aim is to create, import, manipulate and save bitmap (.bmp) files just like paint programs do, only in a command style C++ fashion. The BaX-lib is the main engine of the the BaX-project, which aims to script bitmap files. Using the BaX-lib with tools like mogrify and ffmpeg, it is possible to make videos of sequences of bitmaps. The BaX-library is very useful for making diagrams, doing scientific research and fun art projects.

Using BaX-lib

Compiling and installing the BaX-lib (on Linux) is easily done, in the BaX-lib folder type:

```
[User]$ make
[root]# make install
```

You may want to edit the `Makefile` in order to satisfy your local system.

Incorporating the BaX-lib in your own C++ project is easy and can be done in several ways:
The easiest way is to `#include <BaX.h>` to your header. It is also possible to link to BaX after compilation (and installation) by adding `#include <libBaX.h>` instead, and compiling with the `-lBaX` option or adding `libBaX.o` to your project.

Look at `test.cc` and the `Makefile` for an illustration. You can compile `test.cc` by running

```
[User]$ make test
```

or `[User]$ make ltest` (linked)
after compilation and installation.

Look at `ising.cc` and the `Makefile` for an example program. You can compile `ising.cc` by running

```
[User]$ make ising
```

or `[User]$ make lising` (linked)
after compilation and installation.
You will need mogrify and ffmpeg to run the Ising program see the `ising.cc` for more information.

Types

The BaX-lib uses two data-types: `color` and `bitmap`. Both will be discussed briefly here, members will be discussed in the reference.

`Color` is a 3 byte or 24 bit data-type containing a 24 bit RGB (red, blue, green) value, as is conventional for 24 bit bitmaps. Red, green and blue are all represented by 1 byte or 8 bits and can therefore vary from decimal value 0 to 255. There are also preset colors enlisted in the reference.

`Bitmap` is an arbitrarily sized bitmap container which contains the raw bitmap data (composed of 24 bit colors) as well as information such as the width, hight and size of the bitmap. The bitmap is represented as a 2d grid, with individual pixels being represented by their coordinates (width,height), starting with 0,0 in the top left corner and going until (width-1),(height-1).

Reference
In the file `test.cc` almost all `bitmap` functions are tested.

`struct color :` (let `M` be a `color`)

`r`	red color value	`M.r`	red color value of M
`g`	green color value	`M.g`	green color value of M
`b`	blue color value	`M.b`	blue color value of M

Constructors:

`color M`	M is a color and is initialized as black, meaning `r=0 g=0 b=0`
`color M(1,2,3)`	M is a color and is initialized as `r=1 g=2 b=3`
`color M(16777215)`	M is a color and is initialized as white, meaning `r=255 g=255 b=255`
`color M(0xFFFFFF)`	M is a color and is initialized as white, meaning `r=255 g=255 b=255`

Operators: (let `M` and `N` be a `color`)

`M==N`	A Boolean test to see whether the M is equal to N
`M!=N`	A Boolean test to see whether the M is not equal to N
`!M`	Returns the complementary color of M
`M/N`	Returns the complementary color of M compared to N

Member-functions:

`(int)`	`M.chex()`	return decimal value of M
`(void)`	`M.set(1,2,3)`	set M to `r=1 g=2 b=3`
`(void)`	`M.set(16777215)`	set M to white meaning `r=255 g=255 b=255`
`(void)`	`M.set(0xFFFFFF)`	set M to white meaning `r=255 g=255 b=255`

Numbers in color are unsigned integers `(unsigned int)`.
For all preset colors, read further.

`class bitmap :` (let `bmp` be a `bitmap`) (we will not discuss private members)
`long double pi=3.14159265358979` the number π (the same for every bitmap)
`unsigned long int foutd` the number of files saved from bitmap

Constructors: (let `Q` be a `bitmap`)
`bitmap bmp` bmp is an arbitrarily initialized bitmap (usually completely black)
`bitmap bmp(200,300)` bmp is a 200x300 (in integers) pixels arbitrarily initialized bitmap
`bitmap bmp("xx.bmp")` xx.bmp is loaded from the present directory
`bitmap bmp(Q)` bmp is initialized as the bitmap Q

Member functions: (let `Q` be a `bitmap`, let `x,y,fat` be `long int`, let `pix,pixsh` be `color`)

`(bool) Q.prtct(x,y);`
Check if 0<x<width 0<y<height of Q
`(void) Q.protect(&x,&y);`
Check if 0<x<width 0<y<height of Q else project on Q using periodic up to 1 period boundary conditions
`(color) Q.upget(x,y);`
Obtain the color of pixel x,y in Q without protecting x,y (warning: this function may cause segfaults if if 0<x<width 0<y<height is false)
`(bool) Q.upset(x,y,pix);` (returns false if unable to set)
Set the color of pixel x,y in Q without protecting x,y (warning: this function may cause segfaults if if 0<x<width 0<y<height is false)
`(color) Q.get(x,y);`
Get the color of pixel x,y in Q with protecting x,y
`(bool) Q.set(x,y,pix);` (returns false if unable to set)
`(void) Q.set(fat,x,y,pix);`
`(void) Q.set(fat,x,y,pix,pixsh);`
Change the color of pixel x,y in Q into pix with protecting x,y
fat makes the square pencil biger
pixsh changes the color of the fat pixels
`(color) Q.subget(long int binx,long int biny,x,y);`
Get the average color of all pixel in a square with area binx,biny starting at x,y in Q with protecting x,y
`(void) Q.subset(long int binx,long int biny,x,y,pix);`
Change the color of all pixel in a square with area binx,biny starting at x,y in Q with protecting x,y
`(long int) Q.gcol(x,y);`
Get the integer value of the color of pixel x,y in Q with protecting x,y
`(long int) Q.upgcol(x,y);`
Get the integer value of the color of pixel x,y in Q without protecting x,y
`(void) Q.chcol(color in,color out);`
Change color in into the color out for all pixels in Q
`(void) Q.fillup(pix);`
Change color of all pixels into pix in Q
`(void) Q.csnow(color *val,long int max);`
For all pixels in Q pick a random color from the color array val (with length max) and change it
`(void) Q.snow(color *val,long int max);`
For all pixels in without a color from the color array val (with length max) Q change its color randomly

```
(void) Q.flipx();
```
Flip the x axis of Q (flip vertically)
```
(void) Q.flipy();
```
Flip the y axis of Q (flip horizontally)
```
(void) Q.rotaclock();
```
If Q is square (width equals height), rotate anti-clockwise (in the mathematically positive direction)
```
(void) Q.rotclock();
```
If Q is square (width equals height) rotate clockwise (in the mathematically negative direction)
```
(void) Q.rotate(long double t);
```
If Q is square (width equals height), rotate t (π-radians) (as this operation is protected and images are always square this operation is not well defined)
```
(void) Q.negative();
```
Make Q negative
```
(void) Q.negative(pix);
```
Make Q negative with respect to pix
```
(void) Q.swap(x1,y1,x2,y2);
```
Swap color of pixel x1,y1 with color of pixel x2,y2
```
(void) Q.connex(x1,y1,x2,y2,pix);
(void) Q.connex(fat,x1,y1,x2,y2,pix);
(void) Q.connex(fat,x1,y1,x2,y2,pix,pixsh);
```
Draw a straight line from pixel x1,y1 to pixel x2,y2 with color pix
fat changes the fatness of the line
pixsh changes the color of the fat pixels
```
(void) Q.box(x1,y1,x2,y2,pix);
(void) Q.box(fat,x1,y1,x2,y2,pix);
(void) Q.box(fat,x1,y1,x2,y2,pix,pixsh);
```
Draw a rectangular box from pixel x1,y1 to pixel x2,y2 with color pix
```
(void) Q.vector(x,y,long double r,long double theta,pix);
(void) Q.vector(fat,x,y,long double r,long double theta,pix);
(void) Q.vector(fat,x,y,long double r,long double theta,pix,pixsh);
```
Draw a straight line starting at x,y length r and angle theta (π-radians) with color pix
```
(void) Q.kykel(long double r,long int x,long int y,color pix);
(void) Q.kykel(fat,long double r,x,y,pix);
(void) Q.kykel(fat,long double r,x,y,pix,pixsh);
```
Draw a circle with center x,y and radius r with color pix
```
(void) Q.elkyk(long double rx,long double ry,x,y,pix);
(void) Q.elkyk(fat,long double rx,long double ry,x,y,pix);
(void) Q.elkyk(fat,long double rx,long double ry,x,y,pix,pixsh);
```
Draw an ellipse with center x,y and x radius rx and y radius ry with color pix
```
(void) Q.pc(*x,*y,long int length,pix);
(void) Q.pc(fat,*x,*y,long int length,pix);
(void) Q.pc(fat,*x,*y,long int length,pix,pixsh);
```
Draw a straight line between points given in array *x,*y (with length length) with color pix
```
(void) Q.flood4(x,y,color pix);
```
Flood Q starting from x,y using 4 nearest neighbors

```
(void) Q.flood8(x,y,pix);
```
Flood Q starting from x,y using the 8 nearest neighbors
```
(void) Q.dflood4(x,y,pix);
```
Flood all colors unequal to pix starting at x,y using the 4 nearest neighbors
```
(void) Q.dflood8(x,y,pix);
```
Flood all colors unequal to pix starting at x,y using the 8 nearest neighbors
```
(void) Q.dflood4(x,y,*pix,long int n);
```
Flood all colors that aren't in *pix (with length n) starting at x,y using the 4 nearest neighbors into pix[0]
```
(void) Q.dflood8(x,y,*pix,long int n);
```
Flood all colors that aren't in *pix (with length n) starting at x,y using the 8 nearest neighbors into pix[0]
```
(bool) Q.compare(bitmap *bmp);
```
Check whether Q is equal to bmp
```
(bool) Q.import(bitmap *bmp);
```
Change Q into bmp if Q has the same sizes as bmp (returns true if success)
```
(void) Q.import(x,y,bitmap *bmp);
```
Change a part of Q into bmp starting at x,y
```
(void) Q.import(x,y,bitmap *bmp,*pix,long int n);
```
Change a part of Q into bmp starting at x,y while omitting the colors in pix (length n) in bmp
```
(bool) Q.save(char *fname);
```
(returns true if success)
Save Q as a bitmap with filename (cstring) *fname (don't forget the extension ".bmp")
```
(bool) Q.save_wx(char *fname);
```
(returns true if success)
Save Q as a bitmap with filename (cstring) *fname while automatically setting the extension
```
(bool) Q.save_wx_seq(char *fname);
```
(returns true if success)
Save Q with extension, between the filename and the extension a sequential number is put (this is convenient while making videos with ffmpeg,mencoder or some other video encoder)
```
(char *)Q.fname();
```
Return cstring with the filename of the previously saved Q if anything is saved else returns NULL
```
(void) Q.uitole32(char *le32,long int a);
```
Convert long int to little endian 32bit (yes, this should be an int on most systems)
```
(long int) Q.le32touit(char *le32);
```
Convert little endian 32bit to long int (yes, this should be an int on most systems)
```
(char *)Q.itoa(unsigned long int i);
```
Convert integer to cstring
```
(void) Q.randinit();
```
Seed the random generator used for csnow and snow, (same as srand(time(NULL));)
```
(long int) Q.cwsize();
```
Get width of Q
```
(long int) Q.chsize();
```
Get height of Q
```
(long int) Q.csize();
```
Get filesize of Q
```
(long int) Q.crawsize();
```
Get bitmap size (raw size) of Q
```
(bool) Q.craw(unsigned char *buffer,long int size);
```
(returns true if success)
Get raw bitmap without header of Q

Preset `color`:

```
const color BLACK(0x000000);
const color DARKBLUE(0x00008B);
const color BLUE(0x0000FF);
const color GREEN(0x008000);
const color DARKCYAN(0x008B8B);
const color DARKTURQUOISE(0x00CED1);
const color LIME(0x00FF00);
const color AQUA(0x00FFFF);
const color MIDNIGHTBLUE(0x191970);
const color LIGHTSEAGREEN(0x20B2AA);
const color SEAGREEN(0x2E8B57);
const color LIMEGREEN(0x32CD32);
const color TURQUOISE(0x40E0D0);
const color STEELBLUE(0x4682B4);
const color MEDIUMTURQUOISE(0x48D1CC);
const color DARKOLIVEGREEN(0x556B2F);
const color CORNFLOWERBLUE(0x6495ED);
const color DIMGRAY(0x696969);
const color OLIVEDRAB(0x6B8E23);
const color LIGHTSLATEGRAY(0x778899);
const color LAWNGREEN(0x7CFC00);
const color AQUAMARINE(0x7FFFD4);
const color PURPLE(0x800080);
const color GRAY(0x808080);
const color LIGHTSKYBLUE(0x87CEFA);
const color DARKRED(0x8B0000);
const color SADDLEBROWN(0x8B4513);
const color LIGHTGREEN(0x90EE90);
const color DARKVIOLET(0x9400D3);
const color DARKORCHID(0x9932CC);
const color SIENNA(0xA0522D);
const color DARKGRAY(0xA9A9A9);
const color GREENYELLOW(0xADFF2F);
const color LIGHTSTEELBLUE(0xB0C4DE);
const color FIREBRICK(0xB22222);
const color MEDIUMORCHID(0xBA55D3);
const color DARKKHAKI(0xBDB76B);
const color MEDIUMVIOLETRED(0xC71585);
const color PERU(0xCD853F);
const color TAN(0xD2B48C);
const color PALEVIOLETRED(0xD87093);
const color ORCHID(0xDA70D6);
const color CRIMSON(0xDC143C);
const color PLUM(0xDDA0DD);
const color LIGHTCYAN(0xE0FFFF);
const color DARKSALMON(0xE9967A);
const color PALEGOLDENROD(0xEEE8AA);
const color KHAKI(0xF0E68C);
const color HONEYDEW(0xF0FFF0);
const color SANDYBROWN(0xF4A460);
const color BEIGE(0xF5F5DC);
const color MINTCREAM(0xF5FFFA);
const color SALMON(0xFA8072);
const color LINEN(0xFAF0E6);
const color OLDLACE(0xFDF5E6);
const color FUCHSIA(0xFF00FF);
const color DEEPPINK(0xFF1493);
const color TOMATO(0xFF6347);
const color CORAL(0xFF7F50);
const color LIGHTSALMON(0xFFA07A);
const color LIGHTPINK(0xFFB6C1);
const color GOLD(0xFFD700);
const color NAVAJOWHITE(0xFFDEAD);
const color BISQUE(0xFFE4C4);
const color BLANCHEDALMOND(0xFFEBCD);
const color LAVENDERBLUSH(0xFFF0F5);
const color CORNSILK(0xFFF8DC);
const color FLORALWHITE(0xFFFAF0);
const color YELLOW(0xFFFF00);
const color IVORY(0xFFFFF0);
const color NAVY(0x000080);
const color MEDIUMBLUE(0x0000CD);
const color DARKGREEN(0x006400);
const color TEAL(0x008080);
const color DEEPSKYBLUE(0x00BFFF);
const color MEDIUMSPRINGGREEN(0x00FA9A);
const color SPRINGGREEN(0x00FF7F);
const color CYAN(0x00FFFF);
const color DODGERBLUE(0x1E90FF);
const color FORESTGREEN(0x228B22);
const color DARKSLATEGRAY(0x2F4F4F);
const color MEDIUMSEAGREEN(0x3CB371);
const color ROYALBLUE(0x4169E1);
const color DARKSLATEBLUE(0x483D8B);
const color INDIGO(0x4B0082);
const color CADETBLUE(0x5F9EA0);
const color MEDIUMAQUAMARINE(0x66CDAA);
const color SLATEBLUE(0x6A5ACD);
const color SLATEGRAY(0x708090);
const color MEDIUMSLATEBLUE(0x7B68EE);
const color CHARTREUSE(0x7FFF00);
const color MAROON(0x800000);
const color OLIVE(0x808000);
const color SKYBLUE(0x87CEEB);
const color BLUEVIOLET(0x8A2BE2);
const color DARKMAGENTA(0x8B008B);
const color DARKSEAGREEN(0x8FBC8F);
const color MEDIUMPURPLE(0x9370D8);
const color PALEGREEN(0x98FB98);
const color YELLOWGREEN(0x9ACD32);
const color BROWN(0xA52A2A);
const color LIGHTBLUE(0xADD8E6);
const color PALETURQUOISE(0xAFEEEE);
const color POWDERBLUE(0xB0E0E6);
const color DARKGOLDENROD(0xB8860B);
const color ROSYBROWN(0xBC8F8F);
const color SILVER(0xC0C0C0);
const color INDIANRED(0xCD5C5C);
const color CHOCOLATE(0xD2691E);
const color LIGHTGREY(0xD3D3D3);
const color THISTLE(0xD8BFD8);
const color GOLDENROD(0xDAA520);
const color GAINSBORO(0xDCDCDC);
const color BURLYWOOD(0xDEB887);
const color LAVENDER(0xE6E6FA);
const color VIOLET(0xEE82EE);
const color LIGHTCORAL(0xF08080);
const color ALICEBLUE(0xF0F8FF);
const color AZURE(0xF0FFFF);
const color WHEAT(0xF5DEB3);
const color WHITESMOKE(0xF5F5F5);
const color GHOSTWHITE(0xF8F8FF);
const color ANTIQUEWHITE(0xFAEBD7);
const color LIGHTGOLDENRODYELLOW(0xFAFAD2);
const color RED(0xFF0000);
const color MAGENTA(0xFF00FF);
const color ORANGERED(0xFF4500);
const color HOTPINK(0xFF69B4);
const color DARKORANGE(0xFF8C00);
const color ORANGE(0xFFA500);
const color PINK(0xFFC0CB);
const color PEACHPUFF(0xFFDAB9);
const color MOCCASIN(0xFFE4B5);
const color MISTYROSE(0xFFE4E1);
const color PAPAYAWHIP(0xFFEFD5);
const color SEASHELL(0xFFF5EE);
const color LEMONCHIFFON(0xFFFACD);
const color SNOW(0xFFFAFA);
const color LIGHTYELLOW(0xFFFFE0);
const color WHITE(0xFFFFFF);
```

Examples
See `test.cc` and `ising.cc`
Also check https://www.youtube.com/watch?v=DuVRK0oKgmY
and https://www.youtube.com/watch?v=A8ICCv2a4OM

FAQ
Q: Your programs sucks. It's inefficient and slow.
A: Probably. It's still fun and it can still flood your HDD within a minute on an average computer.

Q: Why did you use long int instead of int?
A: When you're creating huge bitmaps it sometimes is beneficial. At some point, I decided not check whether I can use an int and just went with a long int, on modern hardware it barely matters anyway.

Q: Why can't I save as .jpeg?
A: I wanted to keep the project small. There are thousands of converters out there (my favorite is mogrify) that can do that much better than I can. The aim was to make a fast, lightweight library that uses no special resources. I think I kind of succeeded.

Q: I like your project! Can I help?
A: Thanks! :) Sure, help is always welcome, that is one of the reasons it's GPLv3 licensed, see how to contact in the contact section.

Q: I found a bug where can I file it?
A: File it

License
GPLv3 http://www.gnu.org/licenses/gpl-3.0.txt
Check the `LICENSE` file

Contact
You can contact via sourceforge at http://bax.sourceforge.net/ or http://sourceforge.net/projects/bax/
You can contact me at mail2benny+BaX@gmail.com no spam PLEASE :-)

Acknowledgments
I, Binyamin Chai, would like to thank the following people for their help support along the way:

- Strüdel & ETHZ
- “Ibnatfatil”
- Banjoman

Thanks to the free software foundation and all its friends!