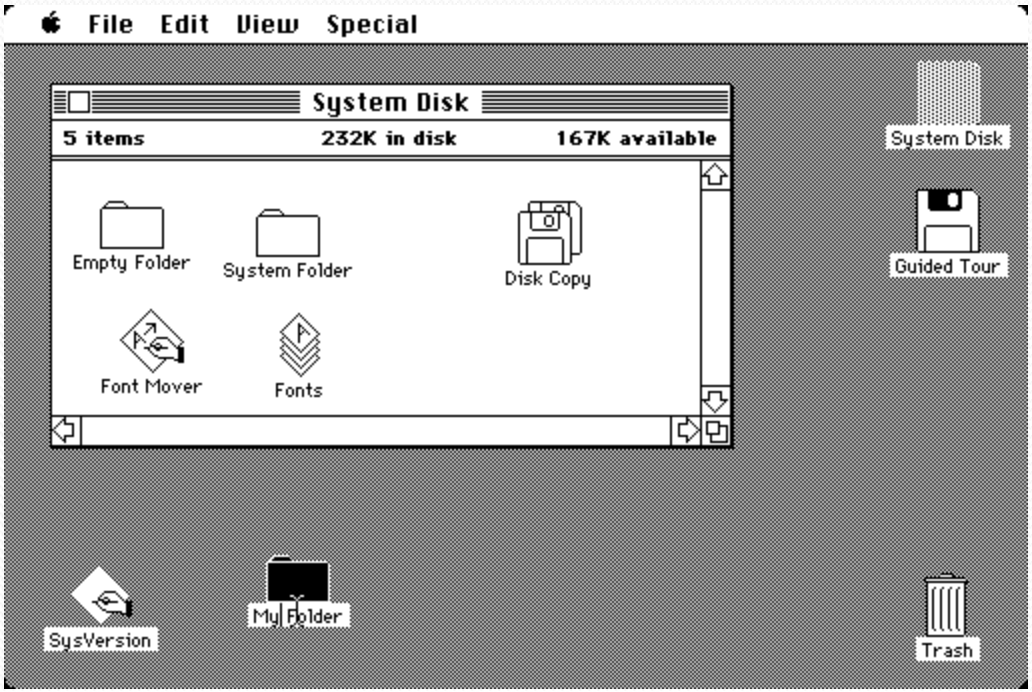
Embedded System
&
Embedded Linux Development
Part 5

Index of today's topic GUI display device

- GUI fundamental
- Linux Framebuffer
- Exercise

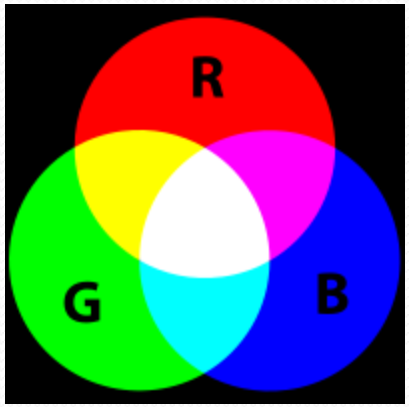
What is GUI

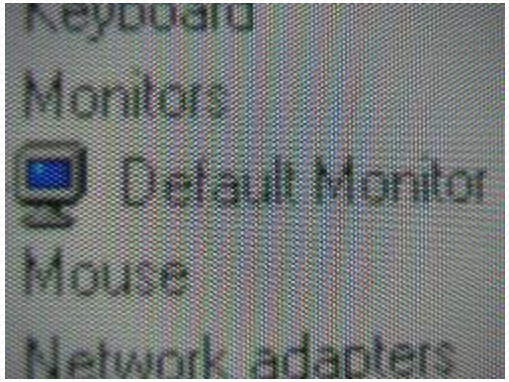
- GUI
  - Graphic User Interface
  - First widely used GUI is Macintosh in 1980s

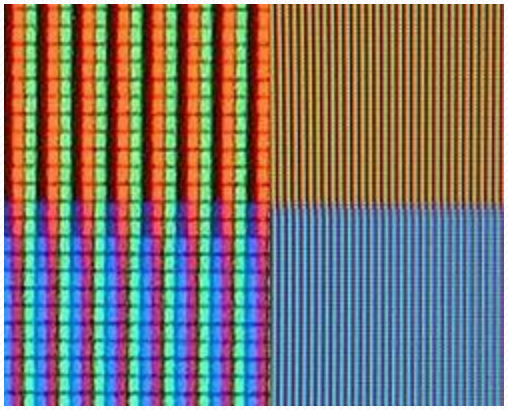


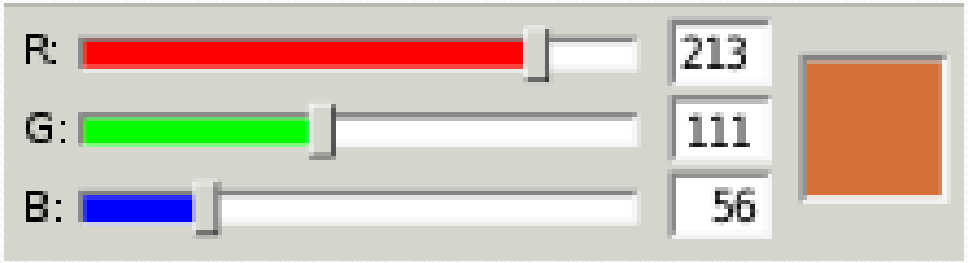
Screen basics

- RGB Color of a pixel
  - Red
  - Green
  - Blue
- True color
  - 8 bit for each channel









Color Depth

- color depth is the number of bit used to represent the color of single pixel.
- also known as bbp (Bit Per Pixel)
- 1-bit – 32 bit
- 2 mode to represent a color
  - Direct Color Mode
  - Indexed Color Mode

Direct Color Digital Representation

- 24 bit color (True Color Mode)

An example of bit groupings in the most common layout of a 24 bit pixel

Sample Length:	8								8								8							
Channel Membership:	**Red**								**Green**								**Blue**							
Bit Number:	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- 16 bit color (High Color Mode)

Sample layout of real 16 bit color data in a 16 bit pixel (in RGBAX notation)

Sample Length:	5					6						5				
Channel Membership:	**Red**					**Green**						**Blue**				
Bit Number:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGBAX Sample Length Notation:	**R. G. B. A. X** **5.6.5.0.0**															

Pixel data in C code

- 24 bit mode
  - #define RGB(R,G,B) ((R<<16)|(G<<8)|B)
  - Example:
    - **unsigned int color;**
    - **color = RGB(255,0,0); //Red**
- 16 bit mode (RGB565)
  - #define RGB(R,G,B) ((R&0xf8)<<8)|((G&0xfc)<<3)|((B&0xf8)>>3)
  - Example:
    - **unsigned short color;**
    - **color = RGB(0, 255, 0); //Green**

Pixel data in C code

- Alpha Channel indicates the opacity of a pixel
- 32 bit mode
  - #define ARGB(A,R,G,B) ((A<<24)|(R<<16)|(G<<8)|B)
  - Example:
    - **unsigned int color;**
    - **color = ARGB(128,255,0,0,); //Red with 50% transparent level**

Indexed Color

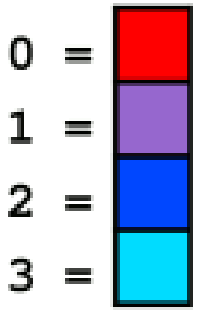
- Color information
  - not directly carried by the image pixel data
  - but stored in a separate piece of data called a palette.
- Image Pixels
  - do not contain the full specification of its color
  - but only its index in the *palette*

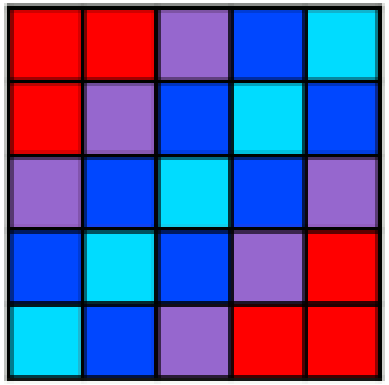
Indexed Color

- Why indexed color is widely used in embedded systems?
  - save lots of memory
    8-bit indexed color image only use 1/3 size of 24-bit true color
  - increase rendering speed

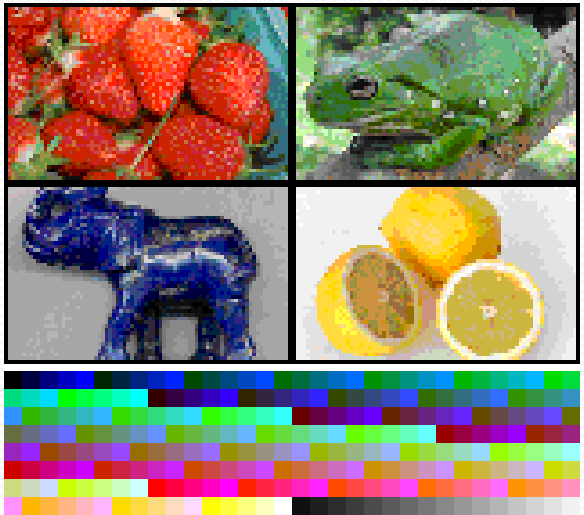
    many hardware support to store palette direct in hardware register

0	0	1	2	3
0	1	2	3	2
1	2	3	2	1
2	3	2	1	0
3	2	1	0	0





Disadvantage of Indexed Color

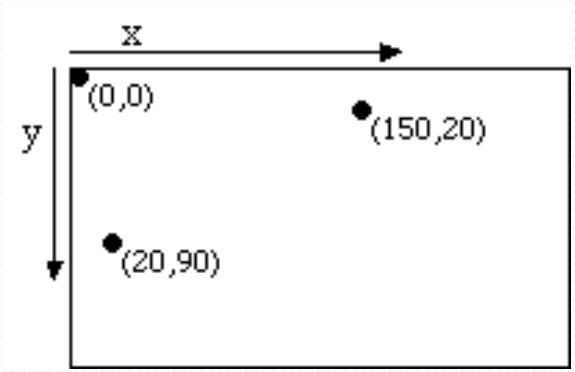


Index of today's topic
GUI display device

- GUI fundamental
- Linux Framebuffer
- Exercise

Framebuffer

- A **framebuffer** is a video output device that drives a video display from a memory buffer containing a complete frame of data



(0,0)	(1,0)	(2,0)	(3,0)	....	(w,0)	(0,1)	(1,1)	(2,1)	...	(x,y)
0	1	2	3	...	w	1*w+0	1*w+1	1*w+2	...	y*w+x

Linux FrameBuffer

- A standard driver in Linux
- Widely used in Embedded systems
- Header file
  - "**linux/fb.h**"
- Open Linux Framebuffer device
  - **fd = open("/dev/fbo", O_RDWR);**

Get screen information

- struct fb_var_screeninfo {
- __u32 **xres**; /* visible resolution*/
- __u32 **yres**;
- __u32 xres_virtual; /* virtual resolution*/
- __u32 yres_virtual;
- __u32 xoffset; /* offset from virtual to visible resolution */
- __u32 yoffset;
- __u32 **bits_per_pixel**; /* bpp*/
- …
- };
- **struct fb_var_screeninfo modeinfo;**
- **ioctl(fd, FBIOGET_VSCREENINFO, &modeinfo);**

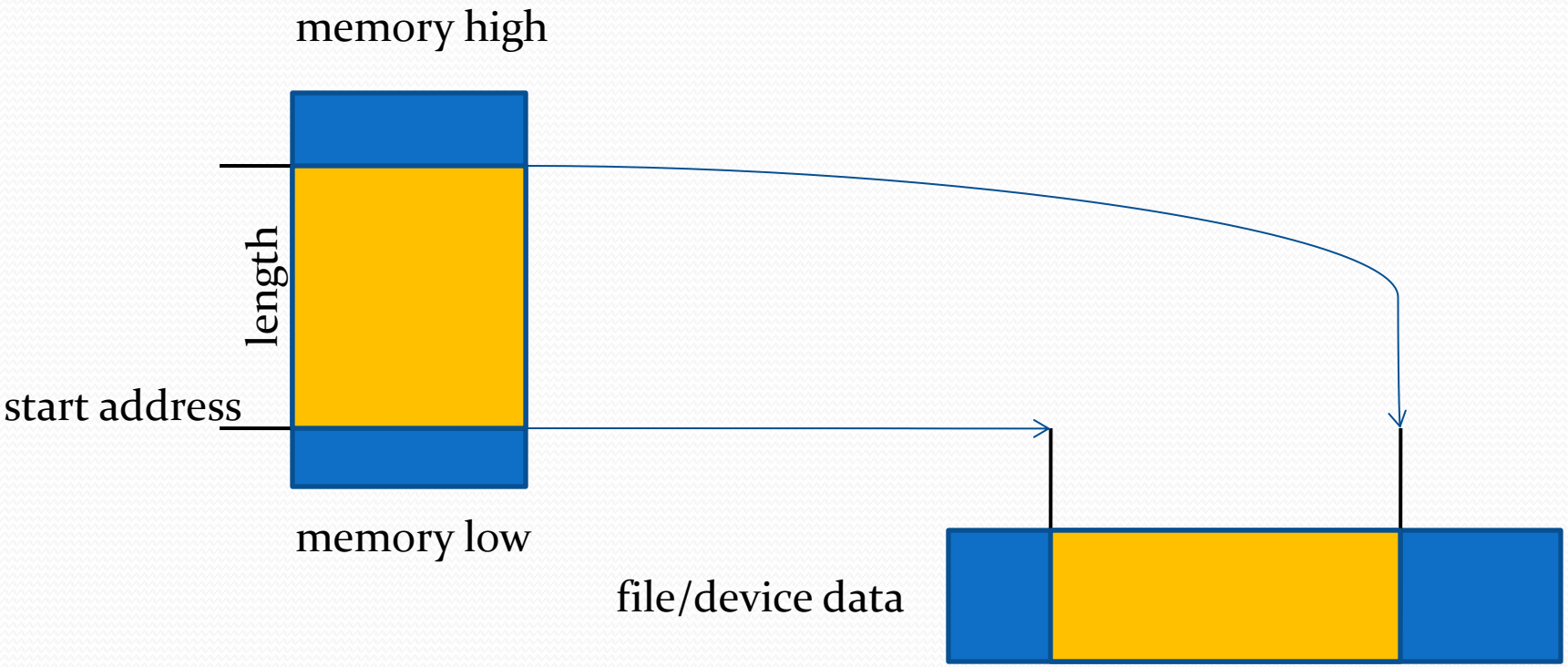
Map Framebuffer to Linux user space memory address

```
char *fbmem_addr;
fbmem_addr = mmap(0, length, PROT_READ |
    PROT_WRITE, MAP_SHARED, fd, 0);
```

- length = screen_width * screen_height * bit_per_pixel/8

mmap()

- #include <sys/mman.h>
- void *mmap(void *addr, size_t len, int prot, int flag, int filedes, off_t off);
- int munmap(void *addr, size_t len);



Paint a RGB565 Pixel

- To paint a point to (x,y) in a screen by a color
  - **color = RGB(r,g,b);**
  - ***(unsigned short*)(fbmem_addr + screen_width*y*bpp/8 + x* bpp/8) = color;**
- or
  - **fbmem_addr[screen_width*y*bbp/8 + x*bbp/8] = color&oxooff;**
  - **fbmem_addr[screen_width*y*bbp/8 + x*bbp/8 + 1] = (color&oxffoo)>>8;**

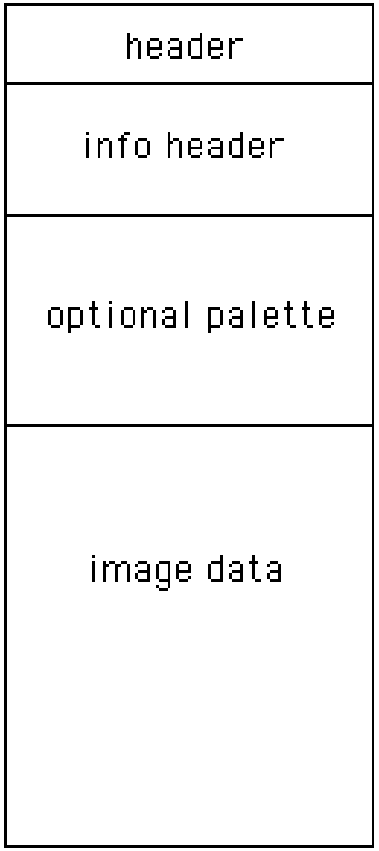
Display Bitmap image

- Load bitmap to memory
  - fopen(), fread()
- draw bitmap pixel to screen line by line
- draw bitmap pixel to screen pixel by pixel

BMP file format

```
#pragma pack(1) /* make sure pack structure by 1 bytes */
typedef struct {
   unsigned short int type;              /* Magic identifier          */
   unsigned int size;                    /* File size in bytes        */
   unsigned short int reserved1, reserved2;
   unsigned int offset;                  /* Offset to image data, bytes */
} HEADER;

typedef struct {
   unsigned int size;              /* Header size in bytes      */
   int width,height;               /* Width and height of image */
   unsigned short int planes;      /* Number of colour planes   */
   unsigned short int bits;        /* Bits per pixel            */
   unsigned int compression;       /* Compression type          */
   unsigned int imagesize;         /* Image size in bytes       */
   int xresolution,yresolution;    /* Pixels per meter          */
   unsigned int ncolours;          /* Number of colours         */
   unsigned int importantcolours;  /* Important colours         */
} INFOHEADER;
```



Render bitmap pixel by pixel

- Get pixel from bitmap (24bit BMP)
  - **data_address = file_start_address + header->offset;**
  - **pixel_address = data_address + y*width*bpp/8 + x;**
- convert it to fit screen format(LCD 16bit RGB565)
  - **unsigned char r,g,b;**
  - **b = pixel_address[0];**
  - **g = pixel_address[1];**
  - **r = pixel_address[2];**
  - **unsigned short color = RGB(r,g,b);**
- draw pixel on screen
  - **draw_pixel(x, y, color);**

Improve Rending Speed

- Use **inline**
- Implement **draw_line**() instead of calling draw_pixel()
- If bitmap is the same color depth with screen, use memcpy() to copy line by line or even block by block(double buffer)
- Clear screen:

```
dd if=/dev/zero  of=/dev/fb0 bs=260k  count=1
```

Exercise 1
Basic painting function

- Implement several painting functions for LCD screen (RGB565 mode)
  - void draw_point(int x, int y, unsigned short color);
  - void draw_line(int x1, int y1, int x2, int y2, unsigned short color);
  - void fill_rect(int x, int y, int width, int height, unsigned short color);
- Clear the screen
- Fill a rectangle in screen: fill_rect(0,0,10,20);
- Move this rect from (0,0) to (200,200) with smooth animation

Exercise 2
Display BMP image on LCD screen

- Use Windows Paint or any other tools to create a 480*272 image, save it in 24bit BMP format
- upload the image to target board
- write a program to
  - open BMP file
  - read the BMP file header
  - read the pixels in BMP data and display it on screen

Exercise 3
Increase rendering speed

- improved rendering speed of image on LCD screen on the basis of the Exercise 2
- Compare Exercise 2 and Exercise 3 , observe the different effect.