

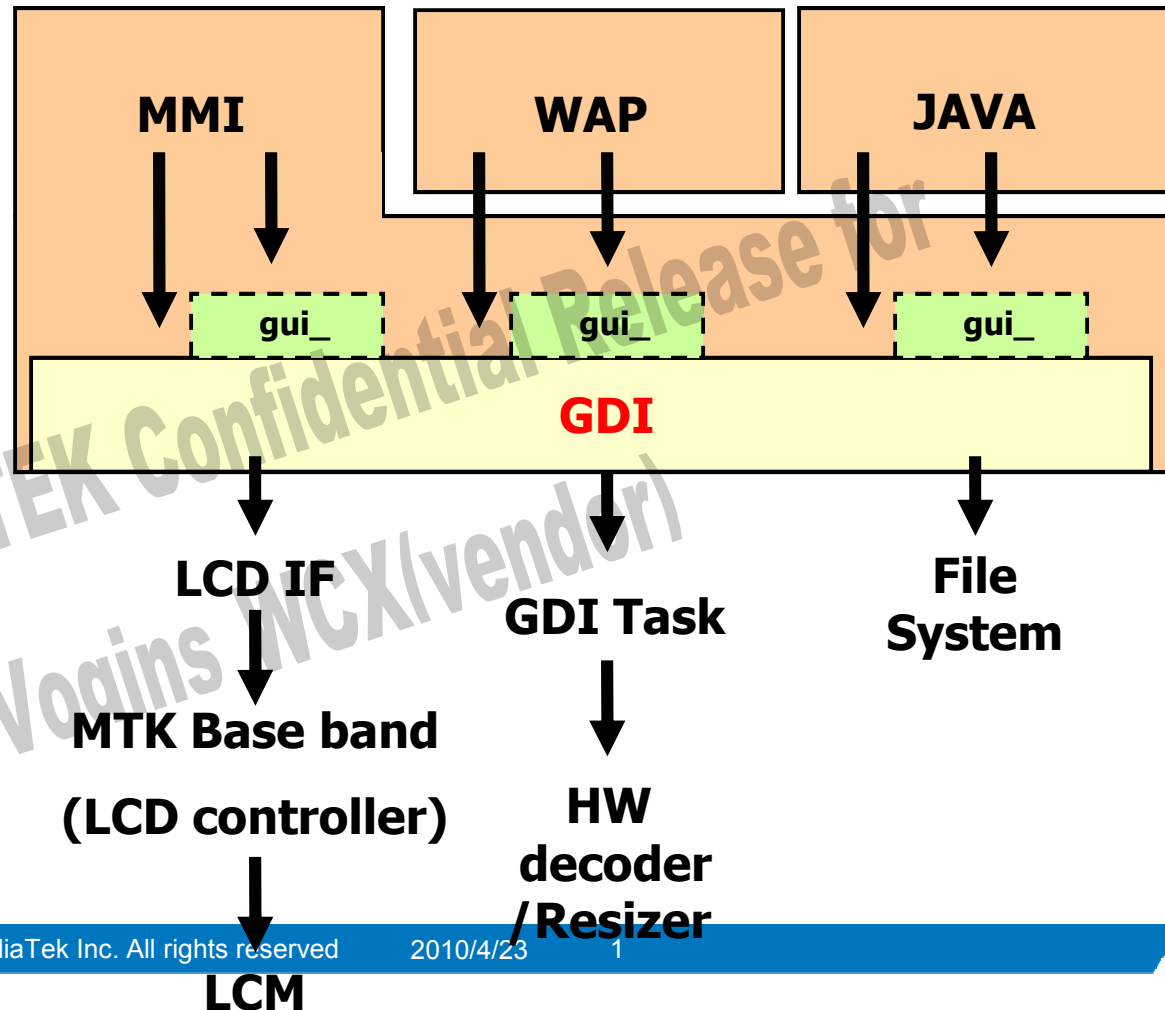


# Graphic Device Interface



# What is GDI

- GDI is a graphic interface to perform drawings



# GDI Features

- Primitive drawing
- Image decoding\drawing
- Image encoding ( currently only support JPG )
- Multi-Layer management
- Multi-LCD management
- Access HW accelerator.
  - ( GIF/ PNG / JPEG / 2D Engine / Resizer )

# GDI Function Naming

- `gdi_layer_create_double_buffer`

↓      ↓  
module    action

- `gdi_anim_draw_id`  
↓  
source

# GDI Overview (1/4)

- GDI use **handle**
  - **gdi\_handle**, **GDI\_HANDLE**
  - Use handles to manipulate GDI object
  - Ex: a layer, an animation gif, a decoding jpeg, etc
- GDI function will return **GDI\_RESULT**
  - Use this result to determine if the request action is succeeded or not
  - Success : **result >= 0**
  - Failed : **result < 0**
  - May find error cause in **gdi\_const.h**
- GDI is **thread safe**
  - GDI function is protected by **MUTEX**

## GDI Overview (2/4)

- GDI Data Type

- typedef U32 gdi\_color;
- typedef S32 gdi\_handle;
- typedef S32 gdi\_result;
  
- #define GDI\_COLOR gdi\_color
- #define GDI\_HANDLE gdi\_handle
- #define GDI\_RESULT gdi\_result

# GDI Overview (3/4)

- GDI is handle based.
- Use handle to control various GDI object.
- Ex:
  - GDI\_RESULT gdi\_layer\_create(... , gdi\_handle \*handle\_ptr);
  - GDI\_RESULT gdi\_layer\_set\_active(gdi\_handle handle);
  - GDI\_RESULT gdi\_layer\_free(gdi\_handle handle);
- Ex:
  - GDI\_RESULT gdi\_image\_draw\_animation(..., gdi\_handle \*handle\_ptr);
  - GDI\_RESULT gdi\_image\_stop\_animation(gdi\_handle a\_handle);
- Ex:
  - GDI\_RESULT gdi\_lcd\_set\_active(gdi\_handle lcd\_handle);
  - GDI\_RESULT gdi\_lcd\_push\_and\_set\_active(gdi\_handle lcd\_handle);

# GDI Overview (4/4)

- Application should include “gdi\_include.h” to use GDI.
- GDI PRIMITIVE
  - Draw point, line, rectangle, etc
  - [gdi\\_primitive.\[c.h\]](#)
- GDI IMAGE
  - Draw gif, bmp, wbmp, jpeg, etc
  - [gdi\\_image.\[c.h\]](#), [gdi\\_image\\_gif.\[c.h\]](#), [gdi\\_image\\_wbmp.\[c.h\]](#), [gdi\\_image\\_bmp.\[c.h\]](#), [gdi\\_image\\_decoder.\[c.h\]](#), [gdi\\_bytestream.\[c.h\]](#) ...
- GDI\_ANIMATE
  - Handle animation playing/ timer control
  - [gdi\\_animate.\[c.h\]](#)
- GDI FONT
  - Draw font
  - [Gdi\\_font.\[c.h\]](#)
- GDI LAYER
  - Multi-Layer management
  - [Gdi\\_layer.\[c.h\]](#)
- GDI LCD
  - Multi-LCD management
  - [Gdi\\_lcd.\[c.h\]](#)
- GDI UTIL
  - Some utility functions
  - [Gdi\\_util.\[c.h\]](#)



# GDI PRIMITIVE (1/4)

- Function
  - Color format transform
  - Draw pixel
  - Draw line, style line
  - Draw rectangle, style rectangle
  - 2D memory copy
- Files
  - gdi\_primitive.c
  - gdi\_primitive.h (interface)
  - gdi\_2d\_engine.c
  - gdi\_2d\_engine.h

# GDI PRIMITIVE (2/4)

- Color Format Transform
  - This two function will convert color format according current active layer color format.
  - **gd\_color\_from\_rgb\_func** gdi\_act\_color\_from\_rgb;
  - **gd\_color\_to\_rgb\_func** gdi\_act\_color\_to\_rgb;
  - Example

```
gdi_color green_start_color;
gdi_color green_end_color;
gdi_color black_color;
U32 A,R,G,B;

// generate GDI_COLOR using A,R,G,B
green_start_color = gdi_act_color_from_rgb(255,30,255,30);
green_end_color = gdi_act_color_from_rgb(255,200,255,200);
black_color = gdi_act_color_from_rgb(255,0,0,0);

// Fetch A,R,G,B color component from GDI_COLOR
gdi_act_color_to_rgb( &A,&R,&G,&B, green_start_color);
```

# GDI PRIMITIVE (3/4)

- Global macro

```
#define GDI_COLOR_WHITE      gdi_act_color_from_rgb(255, 255, 255, 255)
#define GDI_COLOR_BLACK     gdi_act_color_from_rgb(255, 0, 0, 0)
#define GDI_COLOR_GRAY      gdi_act_color_from_rgb(255, 127, 127, 127)
#define GDI_COLOR_RED        gdi_act_color_from_rgb(255, 255, 0, 0)
#define GDI_COLOR_BLUE       gdi_act_color_from_rgb(255, 0, 0, 255)
#define GDI_COLOR_GREEN      gdi_act_color_from_rgb(255, 0, 255, 0)
#define GDI_COLOR_TRANSPARENT gdi_act_color_from_rgb(0, 0, 0, 255)
```

# GDI PRIMITIVE (4/4)

- Draw pixel
  - gdi\_draw\_point
- Draw line & style line
  - gdi\_draw\_line
  - gdi\_draw\_line\_style
- Draw rectangle & style rectangle
  - gdi\_draw\_rect
  - gdi\_draw\_solid\_rect
  - gdi\_draw\_frame\_rect
  - gdi\_draw\_round\_rect
  - gdi\_draw\_button\_rect
  - gdi\_draw\_shadow\_rect
  - gdi\_draw\_gradient\_rect

# GDI IMAGE (1/5)

- Functions
  - Draw static gif, draw gif animation
  - Draw bmp
  - Draw wbmp
  - Draw pbm
  - Draw png
  - Draw jpeg
  - Get image size
  - Get animation frame count
  - Source from files or internal resource
- Files
  - gdi\_image.c
  - **gdi\_image.h (interface)**
  - gdi\_image\_xxxx.c
  - gdi\_image\_xxxx.h

## GDI IMAGE (2/5)

- API naming rule
  - Draw type
  - Source

**gdi\_image\_draw**  
**gdi\_image\_draw\_resized**  
**gdi\_image\_draw\_file**  
**gdi\_image\_draw\_resized\_file**

# GDI IMAGE (3/5)

- Decoder Capability
  - BMP
    - 1bit, 4bit, 8bit, 16bit, 24bit, 32bit
    - RLE (run length encoding , only support from FILE )
  - GIF
    - GIF87, GIF89a
    - Static image or animation
    - Normal and Interlaced mode
  - JPEG
    - SW JPEG do not support progressive mode
  - WBMP
    - Black&white image
  - PNG
    - Do not support alpha color.
  - PBM
    - MTK property format which is old format and replaced by ABM
  - ABM
    - MTK property format which support alpha blending

# GDI IMAGE (4/5)

- APIs

- gdi\_image\_draw (**\_id, \_file**)
- gdi\_image\_draw\_resized
- gdi\_anim\_draw
- gdi\_anim\_draw\_once
- gdi\_anim\_draw\_frames
- gdi\_image\_get\_dimension
- gdi\_anim\_stop( handle)
- gdi\_anim\_stop\_all



# GDI IMAGE (5/5)

- No-Blocking Decoder
  - Process non blocking action in GDI Task ( low priority task)
  - APIs
    - gdi\_image\_nb\_set\_parameter
    - gdi\_image\_nb\_stop
    - gdi\_image\_nb\_stop\_all
    - gdi\_anim\_nb\_set\_parameter
    - gdi\_anim\_nb\_stop
    - gdi\_anim\_nb\_stop\_all
    - gdi\_image\_nb\_draw
    - gdi\_image\_nb\_draw\_resized
    - gdi\_image\_nb\_draw\_file
    - gdi\_image\_nb\_draw\_resized\_file
    - gdi\_anim\_nb\_draw
    - gdi\_anim\_nb\_draw\_resized
    - gdi\_anim\_nb\_draw\_file
    - gdi\_anim\_nb\_draw\_resized\_file

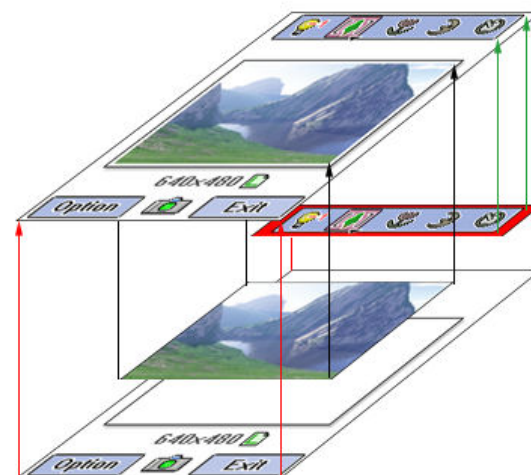
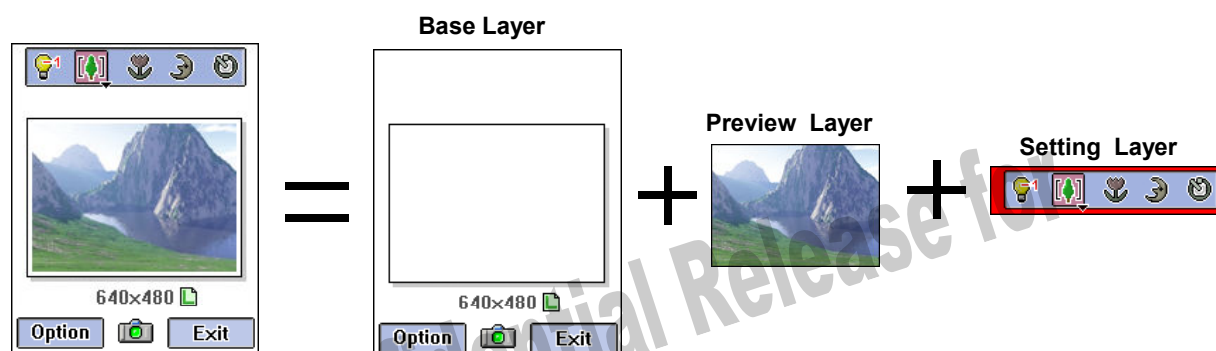
# GDI Font

- Character drawing function
- HW accelerator in MT6219 and after
- Only used by MMI font engine.
- User should use **UI\_**xxx interface to draw text

# GDI\_LAYER (1/4)

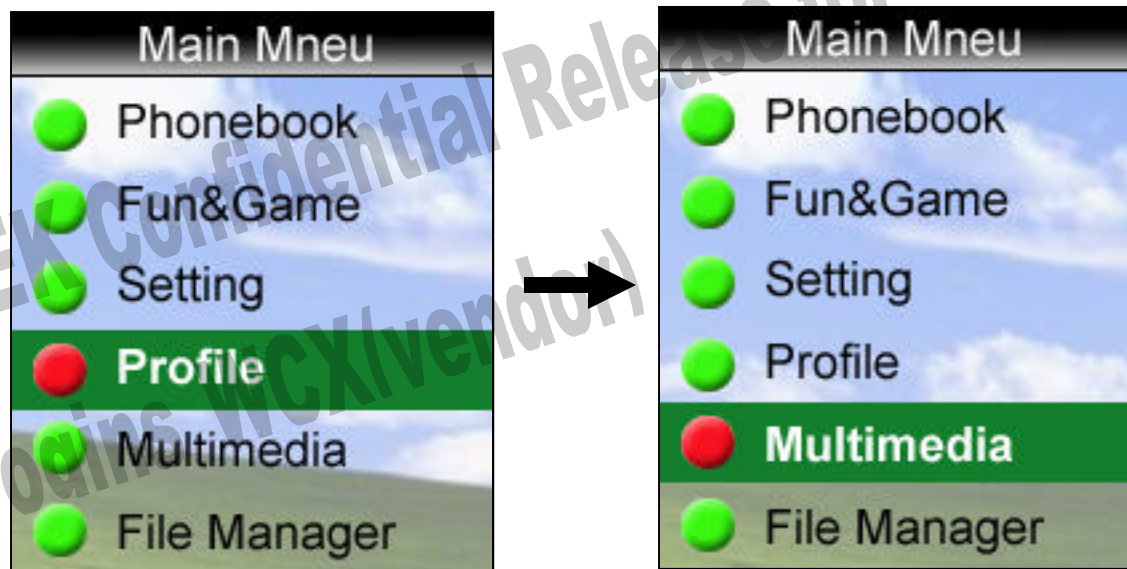
- **HW support 4 layers.** (Merge 4 layer)
  - Chip 6218& 6219 restrict GDI to use 4 layers.
- **GDI support 15 layers.** (Create 15 layers)
  - 15 layers is defined in gdi\_layer.h
  - GDI\_LAYER\_TOTAL\_LAYER\_COUNT
- **How to use**
  - Create layers (no need to create base layer)
  - Draw each layers.
  - Blt layers.

# GDI\_LAYER(2/4)



## GDI LAYER (3/4)

- Drawing a submenu with background image
  - WITHOUT multi-layer
  - **NEED REDRAW EVERYTHING include BACKGROUD**



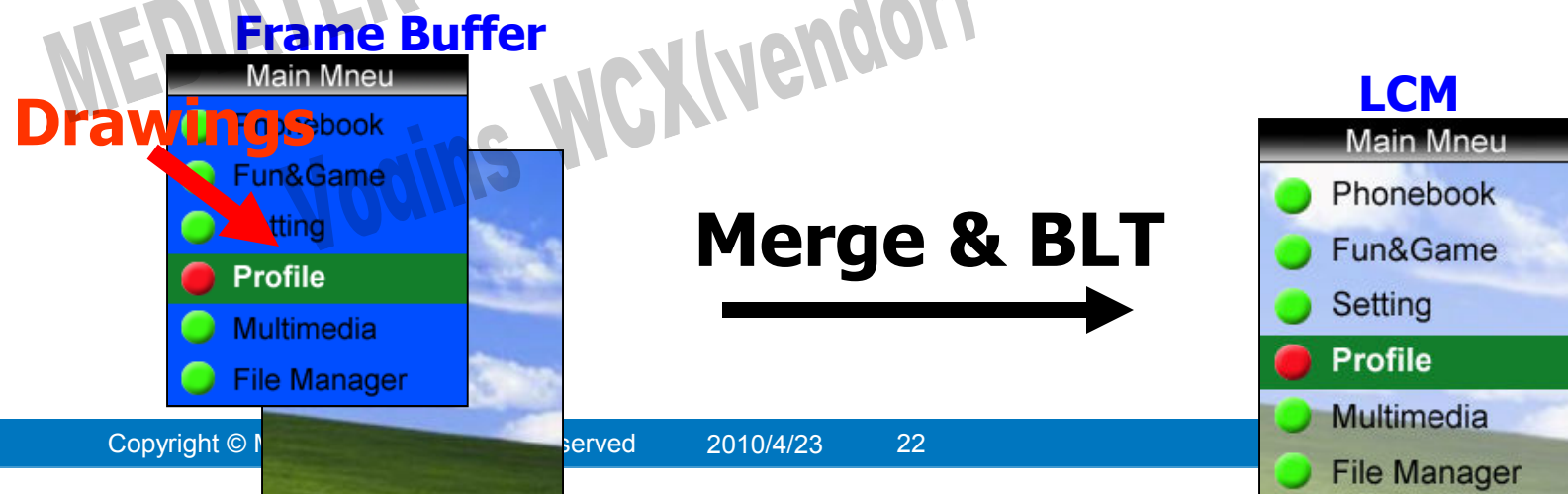
# GDI LAYER (4/4)

- Drawing a submenu with background image
  - WITH multi-layer
  - NO NEED TO REDRAW BACKGROUD



# What is BLT?

- Send Frame buffer data to LCM



# GDI LAYER – Base Layer

- There is always one layer exist
  - Base Layer
  - Same size as LCM size
  - By default – this is the ACTIVE layer
  - All drawings will draw to ACTIVE layer
  - There is a base layer for Main LCD and a base layer for Sub LCD
  - Use gdi\_layer\_get\_active to get base layer's handle
  - Base layer handle
    - GDI\_LAYER\_MAIN\_BASE\_LAYER\_HANDLE
    - GDI\_LAYER\_SUB\_BASE\_LAYER\_HANDLE



# HOW TO USE


- How to use in detail.

- Create each layers. [ [gdi\\_layer\\_create](#) ]
- Before draw on a layer. Set this layer active by calling. [ [gdi\\_layer\\_push\\_and\\_set\\_active](#) ]
- Following drawing function will all draw on this layer.
- After drawing is finished. Restore base layer as active layer. [ [gdi\\_layer\\_pop\\_and\\_restore\\_active](#) ]
- Draw layers to LCD by calling [ [gdi\\_layer\\_blt](#) ]
- Before exit a multi-layer screen, free created layers. [ [gdi\\_layer\\_free](#) ]

## HOW TO USE – Init (1/2)

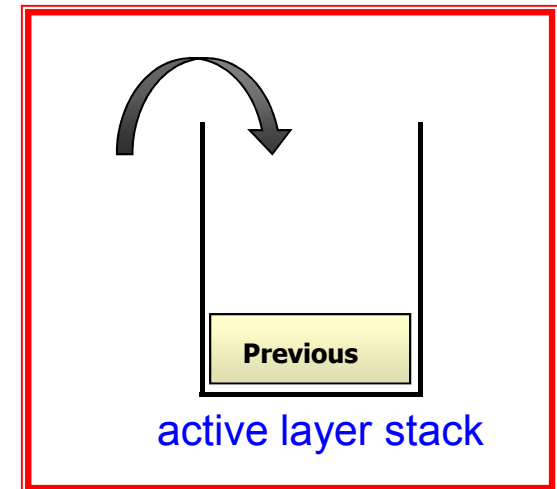
- Create Multi-layer
  - gdi\_layer\_create
  - gdi\_layer\_create\_using\_outside\_memory
  - gdi\_layer\_create\_double
  - gdi\_layer\_create\_double\_using\_outside\_memory
- Create Multi-layer with color format
  - gdi\_layer\_create\_cf
  - gdi\_layer\_create\_cf\_using\_outside\_memory
  - gdi\_layer\_create\_cf\_double
  - gdi\_layer\_create\_cf\_double\_using\_outside\_memory

```
Yc GDI_RESULT gdi_layer_create( S32 x_offset,  
                                S32 y_offset,  
                                S32 width,  
                                S32 height,  
                                gdi_handle *handle_ptr);
```



## HOW TO USE – Init (2/2)

- Set The Layer **ACTIVE**
  - gdi\_layer\_push\_and\_set\_active
    - gdi\_layer\_get\_active - get previous active layer handle
    - gdi\_layer\_set\_active – set this layer active

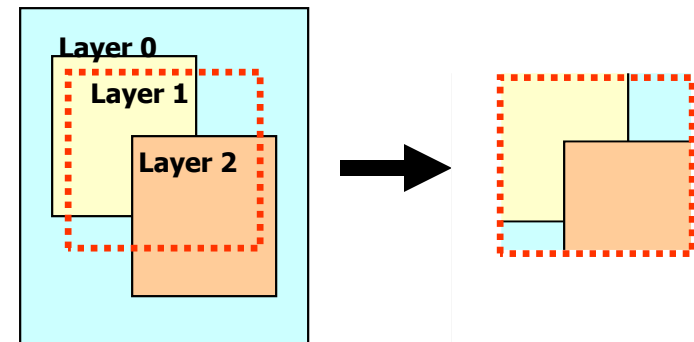


- Set Layer's Property
  - gdi\_layer\_set\_source\_key
    - GDI\_COLOR\_TRANSPARENCY (0, 0, 255)
  - gdi\_layer\_set\_opacity

# HOW TO USE – Drawing

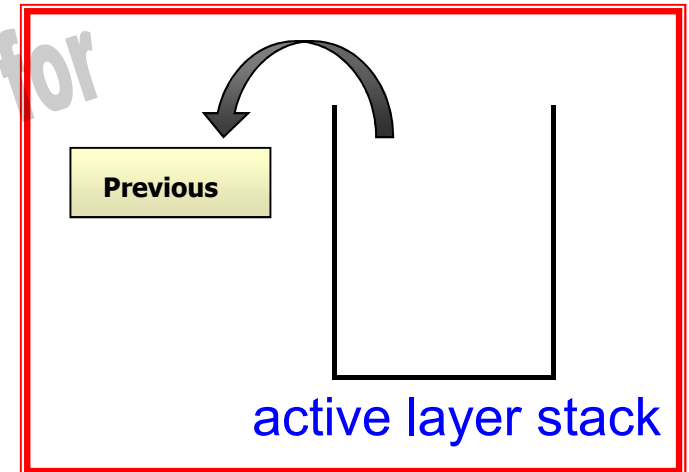
- Input
  - Screen need to redraw
- Drawing
  - Draw line, draw images, etc
- BLT to LCM
  - gdi\_layer\_blt
  - Tell which layers and what region will be blt to LCM
  - gdi\_layer\_set\_blt\_layer
  - gdi\_layer\_blt\_previous

```
gdi_layer_blt( layer0, layer1, layer2, 0,  
              30, 30, 100, 100);
```



# GDI LAYER – DeInit

- Restore to Previous **ACTIVE** Layer
  - gdi\_layer\_pop\_and\_restore\_active
    - gdi\_layer\_set\_active – set previous active layer active again
- Flatten Layers
  - gdi\_layer\_flatten\_to\_base
- Free Layer
  - gdi\_layer\_free
- End Using Multi-layer
  - gdi\_layer\_multi\_layer\_disable



# OTHER GDI LAYER APIs

- Clip Region
  - gdi\_layer\_get\_clip
  - gdi\_layer\_set\_clip
  - gdi\_layer\_push\_clip
  - gdi\_layer\_pop\_clip
  - gdi\_layer\_reset\_clip
  
  - gdi\_layer\_get\_text\_clip
  - gdi\_layer\_set\_text\_clip
  - gdi\_layer\_reset\_text\_clip
  - gdi\_layer\_push\_text\_clip
  - gdi\_layer\_pop\_text\_clip

# Rules! (1/3)

## ■ RULE 1

- All action will take effect on the layer after you set it active.
  - Drawing.
  - Set clip, transparency, source key.
  - Move...etc.

## ■ RULE 2

- Always restore previous layer setting before leaving your application.
  - `gdi_layer_push_and_set_active`
  - `gdi_layer_pop_and_restore_active.`

# Rules! (2/3)

## ■ RULE 3

- Some function should be **PAIRED!**
  - gdi\_layer\_lock\_frame\_buffer
  - gdi\_layer\_unlock\_frame\_buffer
  - gdi\_layer\_push\_clip
  - gdi\_layer\_pop\_clip
  - gdi\_layer\_push\_text\_clip
  - gdi\_layer\_pop\_text\_clip
  - gdi\_layer\_push\_and\_set\_active
  - gdi\_layer\_pop\_and\_restore\_active
- Be careful when return in the middle of a function.



# Rules! (3/3)

## ■ RULE 4

- Each GDI function is protected by recursive mutex.
- You should use GDI\_LOCK / GDI\_UNLOCK to speed up your application.
- Example:

### GDI\_LOCK

```
gdi_layer_lock_frame_buffer  
pixmap_UI_reset_clip  
show_status_icons  
....
```

```
gdi_layer_unlock_frame_buffer  
gdi_layer_blt
```

### GDI\_UNLOCK

# GDI LCD

- Support multiple LCD
- Files
  - gdi\_lcd.c
  - gdi\_lcd.h
- APIs
  - gdi\_lcd\_set\_active
  - gdi\_lcd\_get\_active
  - UI\_set\_sub\_LCD\_graphics\_context()
  - UI\_set\_main\_LCD\_graphics\_context()



## Q & A