图像绘制

- 图像绘制
  - 图像
    - createPng
    - drawBmp
    - drawBmpFile
    - drawBmpUrl
    - drawJpg
    - drawJpgFile
    - drawJpgUrl
    - drawPng
    - drawPngFile
    - drawPngUrl
    - drawQoi
    - drawQoiFile
    - drawQoiUrl
    - pushAlphaImage
    - pushGrayscaleImage
    - pushGrayscaleImageAffine
    - pushGrayscaleImageRotateZoom
    - pushImage
    - pushImageDMA
    - pushImageAffine
    - pushImageAffineWithAA
    - pushImageRotateZoom
    - pushImageRotateZoomWithAA
  - 图像颜色转换
    - setSwapBytes
    - getSwapBytes
    - swap565
    - swap888

图像

createPng

函数原型:

```
void* createPng( size_t* datalen, int32_t x = 0, int32_t y = 0, int32_t width = 0, int32_t height = 0)
```

功能说明:

- 截屏功能，会将面板中显示的数据以PNG格式保存到设备内存中。

> **注意事项:**
>
> 能够存储的图像大小取决于内存容量。例如，如果禁用了 PSRAM 功能，高分辨率高色深的图像将无法被保存。

传入参数:

- datalen：图像数据长度
- x：截图起始点 x 坐标
- y：截图起始点 y 坐标
- w：截图宽度
- h：截图高度

返回值:

- void*:
- 截屏成功时：指向 PNG 图像数据的内存区域的指针
- 截屏失败时：nullptr

案例程序:

cpp

```cpp
#include <Arduino.h>
#include <M5GFX.h>
#include <M5Unified.h>

M5GFX display;
size_t png_datalen = 320 * 240;
uint8_t* PngData = (uint8_t*)malloc(png_datalen * sizeof(uint8_t));

void setup() {
    display.begin();
    display.setColorDepth(8);
    display.setRotation(1);
    display.clear(TFT_WHITE);
    display.setTextFont(&fonts::FreeSansOblique12pt7b);
    display.setTextColor(TFT_BLACK);
    delay(1000);
    uint16_t x = display.width() / 2;
    uint16_t y = display.height() / 2;

    display.drawCenterString("Create PNG Test", x, y);
    delay(2000);
    PngData = (uint8_t*)display.createPng(&png_datalen, 0, 0, 320, 240);
    display.drawCenterString("Screenshot successful", x, y+30);
    delay(2000);
    display.clear(TFT_WHITE);
    delay(1000);
    display.drawCenterString("5s later show screenshot", x, y);
    delay(5000);
    display.drawPng(PngData, png_datalen);
}

void loop() {
}
```

drawBmp

函数原型1：

```
void drawBmp(const uint8_t *data, uint32_t len, int32_t x=0, int32_t y=0, int32_t maxWidth=0, int32_t m
```

函数原型2：

```
void drawBmp(DataWrapper *data, int32_t x=0, int32_t y=0, int32_t maxWidth=0, int32_t maxHeight=0, int3
```

功能说明：

- 绘制/显示 BMP 图像

传入参数：

- data：图像数据指针
- const uint8_t：原始图像数据指针
- DataWrapper：包装的图像数据对象指针 （关于DataWrapper）
- len：数据长度
- x：绘制起点 x 坐标
- y：绘制起点 y 坐标
- maxWidth：图像最大宽度
- maxHeight：图像最大高度
- offX：x 轴偏移量
- offY：y 轴偏移量
- scale_x：x 方向缩放比例
- scale_y：y 方向缩放比例
- datum：图像对齐方式，默认为左上对齐 （d关于atum_t）

返回值：

- null

drawBmpFile

函数原型1：

```
void drawBmpFile(T &fs, const char *path, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t m
```

函数原型3：

```
void drawBmpFile(DataWrapper* file, const char *path, int32_t x = 0, int32_t y = 0, int32_t maxWidth =
```

函数原型2：

```
void drawBmpFile(const char *path, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t maxHeigh
```

功能说明：

- 从 BMP 文件绘制/显示 BMP 图像

> **注意事项：**
>
> 要使用此功能，需在包含 `<M5GFX.h>` 之前写入 `#include <SPIFFS.h>` 或者 `#include <SD.h>` 。

传入参数:

- fs：文件系统对象
- SPIFFS
- SD

  etc
- path：BMP 文件路径
- file：DataWrapper 结构体指针 （关于DataWrapper）
- x：绘制起点 x 坐标
- y：绘制起点 y 坐标
- maxWidth：图像最大宽度
- maxHeight：图像最大高度
- offX：x 轴偏移量
- offY：y 轴偏移量
- scale_x：x 方向缩放比例
- scale_y：y 方向缩放比例
- datum：图像对齐方式，默认为左上对齐 （关于datum_t）

返回值:

- null

drawBmpUrl

函数原型:

```
bool drawBmpUrl(const char* url, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t maxHeight
bool drawBmpUrl(String& url, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t maxHeight = 0,
```

功能说明:

- 从链接绘制/显示 BMP 图像

> **注意事项:**
>
> 要使用此功能，需在包含 `<M5GFX.h>` 之前写入 `#include <HTTPClient.h>` 。

传入参数:

- url：图像链接
- x：绘制起点 x 坐标
- y：绘制起点 y 坐标
- maxWidth：图像最大宽度
- maxHeight：图像最大高度
- offX：x 轴偏移量
- offY：y 轴偏移量
- scale_x：x 方向缩放比例
- scale_y：y 方向缩放比例
- datum：图像对齐方式，默认为左上对齐

返回值:

- bool
- true：绘制/显示成功

- false：绘制/显示失败

drawJpg

函数原型1：

```
void drawJpg(const uint8_t *data, uint32_t len, int32_t x=0, int32_t y=0, int32_t maxWidth=0, int32_t m
```

函数原型2：

```
void drawJpg(DataWrapper *data, int32_t x=0, int32_t y=0, int32_t maxWidth=0, int32_t maxHeight=0, int3
```

功能说明：

- 绘制/显示 JPG 图像

传入参数：

- data
- const uint8_t：原始常量数据指针
- DataWrapper：包装数据对象指针
- len：数据长度
- x：绘制起点 x 坐标
- y：绘制起点 y 坐标
- maxWidth：图像最大宽度
- maxHeight：图像最大高度
- offX：x 轴偏移量
- offY：y 轴偏移量
- scale_x：x 方向缩放比例
- scale_y：y 方向缩放比例
- datum：图像对齐方式，默认为左上对齐 （关于datum_t）

返回值：

- null

drawJpgFile

函数原型1：

```
void drawJpgFile(T &fs, const char *path, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t m
```

函数原型3：

```
void drawJpgFile(DataWrapper* file, const char *path, int32_t x = 0, int32_t y = 0, int32_t maxWidth =
```

函数原型2：

```
void drawJpgFile(const char *path, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t maxHeigh
```

功能说明：

- 从 JPG 文件绘制/显示 JPG 图像

> **注意事项:**
>
> 要使用此功能，需在包含 `<M5GFX.h>` 之前写入 `#include <SPIFFS.h>` 或者 `#include <SD.h>` 。

传入参数:

- fs：文件系统对象
- SPIFFS
- SD

  etc
- path：JPG 文件路径
- file：DataWrapper 结构体指针 （关于DataWrapper）
- x：绘制起点 x 坐标
- y：绘制起点 y 坐标
- maxWidth：图像最大宽度
- maxHeight：图像最大高度
- offX：x 轴偏移量
- offY：y 轴偏移量
- scale_x：x 方向缩放比例
- scale_y：y 方向缩放比例
- datum：图像对齐方式，默认为左上对齐 （关于datum_t）

返回值:

- null

drawJpgUrl

函数原型:

```
bool drawJpgUrl(const char* url, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t maxHeight
bool drawJpgUrl(String& url, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t maxHeight = 0,
```

功能说明:

- 从链接绘制/显示 JPG 图像

> **注意事项:**
>
> 要使用此功能，需在包含 `<M5GFX.h>` 之前写入 `#include <HTTPClient.h>` 。

传入参数:

- url：图像链接
- x：绘制起点 x 坐标
- y：绘制起点 y 坐标
- maxWidth：图像最大宽度
- maxHeight：图像最大高度
- offX：x 轴偏移量
- offY：y 轴偏移量
- scale_x：x 方向缩放比例
- scale_y：y 方向缩放比例
- datum：图像对齐方式，默认为左上对齐

返回值:

- bool
- true：绘制/显示成功
- false：绘制/显示失败

drawPng

函数原型1:

```
void drawPng(const uint8_t *data, uint32_t len, int32_t x=0, int32_t y=0, int32_t maxWidth=0, int32_t m
```

函数原型2:

```
void drawPng(DataWrapper *data, int32_t x=0, int32_t y=0, int32_t maxWidth=0, int32_t maxHeight=0, int3
```

功能说明:

- 绘制/显示 PNG 图像

传入参数:

- data
- const uint8_t：原始常量数据指针
- DataWrapper：包装数据对象指针
- len：数据长度
- x：绘制起点 x 坐标
- y：绘制起点 y 坐标
- maxWidth：图像最大宽度
- maxHeight：图像最大高度
- offX：x 轴偏移量
- offY：y 轴偏移量
- scale_x：x 方向缩放比例
- scale_y：y 方向缩放比例
- datum：图像对齐方式，默认为左上对齐 （关于datum_t）

drawPngFile

函数原型1:

```
void drawPngFile(T &fs, const char *path, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t m
```

函数原型3:

```
void drawPngFile(DataWrapper* file, const char *path, int32_t x = 0, int32_t y = 0, int32_t maxWidth =
```

函数原型2:

```
void drawPngFile(const char *path, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t maxHeigh
```

功能说明:

- 从 PNG 文件绘制/显示 PNG 图像

传入参数:

- fs：文件系统对象
- SPIFFS
- SD

  etc
- path：PNG 文件路径
- file：DataWrapper 结构体指针 （关于DataWrapper）
- x：绘制起点 x 坐标
- y：绘制起点 y 坐标
- maxWidth：图像最大宽度
- maxHeight：图像最大高度
- offX：x 轴偏移量
- offY：y 轴偏移量
- scale_x：x 方向缩放比例
- scale_y：y 方向缩放比例
- datum：图像对齐方式，默认为左上对齐 （关于datum_t）

返回值:

- null

drawPngUrl

函数原型:

```
bool drawPngUrl(const char* url, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t maxHeight
bool drawPngUrl(String& url, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t maxHeight = 0,
```

功能说明:

- 从链接绘制/显示 PNG 图像

> **注意事项:**
>
> 要使用此功能，需在包含 `<M5GFX.h>` 之前写入 `#include <HTTPClient.h>` 。

传入参数:

- url：图像链接
- x：绘制起点 x 坐标
- y：绘制起点 y 坐标
- maxWidth：图像最大宽度
- maxHeight：图像最大高度
- offX：x 轴偏移量
- offY：y 轴偏移量
- scale_x：x 方向缩放比例
- scale_y：y 方向缩放比例
- datum：图像对齐方式，默认为左上对齐

返回值:

- bool
- true：绘制/显示成功

- false：绘制/显示失败

drawQoi

函数原型1：

```
void drawQoi(const uint8_t *data, uint32_t len, int32_t x=0, int32_t y=0, int32_t maxWidth=0, int32_t m
```

函数原型2：

```
void drawQoi(DataWrapper *data, int32_t x=0, int32_t y=0, int32_t maxWidth=0, int32_t maxHeight=0, int3
```

功能说明：

- 绘制/显示 Qoi 图像

传入参数：

- data
- const uint8_t：原始常量数据指针
- DataWrapper：包装数据对象指针
- len：数据长度
- x：绘制起点 x 坐标
- y：绘制起点 y 坐标
- maxWidth：图像最大宽度
- maxHeight：图像最大高度
- offX：x 轴偏移量
- offY：y 轴偏移量
- scale_x：x 方向缩放比例
- scale_y：y 方向缩放比例
- datum：图像对齐方式，默认为左上对齐 （关于datum_t）

返回值：

- null

drawQoiFile

函数原型1：

```
void drawQoiFile(T &fs, const char *path, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t m
```

函数原型3：

```
void drawQoiFile(DataWrapper* file, const char *path, int32_t x = 0, int32_t y = 0, int32_t maxWidth =
```

函数原型2：

```
void drawQoiFile(const char *path, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t maxHeigh
```

功能说明：

- 从 Qoi 文件绘制/显示 Qoi 图像

> **注意事项:**
>
> 要使用此功能，需在包含 `<M5GFX.h>` 之前写入 `#include <SPIFFS.h>` 或者 `#include <SD.h>` 。

传入参数:

- fs：文件系统对象
- SPIFFS
- SD

  etc
- path：Qoi 文件路径
- file：DataWrapper 结构体指针 （关于DataWrapper）
- x：绘制起点 x 坐标
- y：绘制起点 y 坐标
- maxWidth：图像最大宽度
- maxHeight：图像最大高度
- offX：x 轴偏移量
- offY：y 轴偏移量
- scale_x：x 方向缩放比例
- scale_y：y 方向缩放比例
- datum：图像对齐方式，默认为左上对齐 （关于datum_t）

返回值:

- null

drawQoiUrl

函数原型:

```
bool drawQoiUrl(const char* url, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t maxHeight
bool drawQoiUrl(String& url, int32_t x = 0, int32_t y = 0, int32_t maxWidth = 0, int32_t maxHeight = 0,
```

功能说明:

- 从链接绘制/显示 Qoi 图像

> **注意事项:**
>
> 要使用此功能，需在包含 `<M5GFX.h>` 之前写入 `#include <HTTPClient.h>` 。

传入参数:

- url：图像链接
- x：绘制起点 x 坐标
- y：绘制起点 y 坐标
- maxWidth：图像最大宽度
- maxHeight：图像最大高度
- offX：x 轴偏移量
- offY：y 轴偏移量
- scale_x：x 方向缩放比例
- scale_y：y 方向缩放比例
- datum：图像对齐方式，默认为左上对齐

返回值：

- bool
- true：绘制/显示成功
- false：绘制/显示失败

> **说明：**
>
> 本案例程序所用设备为 `M5Core2` ， 请根据实际使用设备更改SD卡控制引脚。

案例程序：

cpp

```cpp
#include <Arduino.h>
#include <SPI.h>
#include <SD.h>
#include <M5Unified.h>
#include <M5GFX.h>

#define SD_SPI_CS_PIN   4
#define SD_SPI_SCK_PIN  18
#define SD_SPI_MISO_PIN 38
#define SD_SPI_MOSI_PIN 23

void setup() {
  M5.begin();

  M5.Display.setTextFont(&fonts::Orbitron_Light_24);
  M5.Display.setTextSize(1);

  // SD Card Initialization
  SPI.begin(SD_SPI_SCK_PIN, SD_SPI_MISO_PIN, SD_SPI_MOSI_PIN, SD_SPI_CS_PIN);
  if (!SD.begin(SD_SPI_CS_PIN, SPI, 25000000)) {
    // Print a message if SD card initialization failed or if the SD card does not exist.
    M5.Display.print("\n SD card not detected\n");
    while (1)
      ;
  } else {
    M5.Display.print("\n SD card detected\n");
  }
  delay(1000);

  M5.Display.print("\n SD card read test...\n");
  if (SD.open("/TestPicture01.png", FILE_READ, false)) {
    M5.Display.print(" PNG file 01 detected\n");
  } else {
    M5.Display.print(" PNG file 01 not detected\n");
  }
  if (SD.open("/TestPicture02.png", FILE_READ, false)) {
    M5.Display.print(" PNG file 02 detected\n");
  } else {
    M5.Display.print(" PNG file 02 not detected\n");
  }
}

void loop() {
  // Read image file and draw picture
  //         drawBmpFile
  //         drawJpgFile
  //         drawQoiFile
  M5.Display.drawPngFile(SD, "/TestPicture01.png");
  delay(1000);
  M5.Display.drawPngFile(SD, "/TestPicture02.png");
```

```
51    delay(1000);
52  }
```

该程序会循环播放 SD 卡中的两张 PNG 图片。

Core2_Arduino_sdcard.mp4

pushAlphaImage

函数原型:

```
void pushAlphaImage(int32_t x, int32_t y, int32_t w, int32_t h, const T* data)
```

功能说明:

- 将带有 alpha 通道的图像推送到显示屏上，仅支持24位及以上的图像格式。

传入参数:

- x：图像左上角的 x 坐标
- y：图像左上角的 y 坐标
- w：图像的宽度
- h：图像的高度
- param：指向 pixelcopy_t 结构体的指针，包含图像数据和其他参数
- data：指向图像像素数据的指针

返回值:

- null

pushGrayscaleImage

函数原型1:

```
void pushGrayscaleImage(int32_t x, int32_t y, int32_t w, int32_t h, const uint8_t* image, color_depth_t
```

功能说明:

- 将灰度图像推送到显示屏上

传入参数:

- x：图像左上角的 x 坐标
- y：图像左上角的 y 坐标
- w：图像的宽度
- h：图像的高度
- image：指向图像数据的指针
- depth：图像的颜色深度
- forecolor：前景色
- backcolor：背景色

返回值:

- null

pushGrayscaleImageAffine

函数原型:

```
void pushGrayscaleImageAffine(const float matrix[6], int32_t w, int32_t h, const uint8_t* image, color_
```

功能说明：

- 将灰度图像以仿射变换的方式推送到显示屏上

传入参数：

- matrix：6 个浮点数的数组，表示仿射变换矩阵
- w：图像的宽度
- h：图像的高度
- image：指向图像数据的指针
- depth：图像的颜色深度
- forecolor：前景色
- backcolor：背景色

返回值：

- null

pushGrayscaleImageRotateZoom

函数原型：

```
void pushGrayscaleImageRotateZoom(float dst_x, float dst_y, float src_x, float src_y, float angle, floa
```

功能说明：

- 将灰度图像以仿射变换的方式推送到显示屏上

传入参数：

- dst_x：目标图像左上角的 x 坐标
- dst_y：目标图像左上角的 y 坐标
- src_x：源图像左上角的 x 坐标
- src_y：源图像左上角的 y 坐标
- angle：旋转角度（以弧度为单位）
- zoom_x：x 方向的缩放比例
- zoom_y：y 方向的缩放比例
- w：图像的宽度
- h：图像的高度
- image：指向图像数据的指针
- depth：图像的颜色深度
- forecolor：前景色
- backcolor：背景色

返回值：

- null

pushImage

函数原型1：

```
void pushImage(int32_t x, int32_t y, int32_t w, int32_t h, const T* data)
```

函数原型2：

```
void pushImage(int32_t x, int32_t y, int32_t w, int32_t h, const T1* data, const T2& transparent)
```

函数原型3：

```
void pushImage(int32_t x, int32_t y, int32_t w, int32_t h, const void* data, color_depth_t depth, const
```

函数原型4：

```
void pushImage(int32_t x, int32_t y, int32_t w, int32_t h, const void* data, uint32_t transparent, colo
```

功能说明：

- 将图像推送到显示屏上

传入参数：

- x：图像左上角的 x 坐标
- y：图像左上角的 y 坐标
- w：图像的宽度
- h：图像的高度
- data：指向图像像素数据的指针
- transparent：透明色
- depth：图像的颜色深度
- palette：调色板指针

返回值：

- null

pushImageDMA

函数原型1：

```
void pushImageDMA(int32_t x, int32_t y, int32_t w, int32_t h, const T* data)
```

函数原型2：

```
void pushImageDMA(int32_t x, int32_t y, int32_t w, int32_t h, const void* data, color_depth_t depth, co
```

功能说明：

- 使用 DMA 将图像推送到显示屏上

传入参数：

- x：图像左上角的 x 坐标
- y：图像左上角的 y 坐标
- w：图像的宽度
- h：图像的高度
- data：指向图像像素数据的指针
- depth：图像的颜色深度
- palette：调色板指针

返回值:

- null

pushImageAffine

函数原型1:

```
void pushImageAffine(const float matrix[6], int32_t w, int32_t h, const T* data)
```

函数原型2:

```
void pushImageAffine(const float matrix[6], int32_t w, int32_t h, const T1* data, const T2& transparent
```

函数原型3:

```
void pushImageAffine(const float matrix[6], int32_t w, int32_t h, const void* data, color_depth_t depth
```

函数原型4:

```
void pushImageAffine(const float matrix[6], int32_t w, int32_t h, const void* data, uint32_t transparen
```

功能说明:

- 使用仿射变换将图像推送到显示屏上

传入参数:

- matrix：6 个浮点数的数组，表示仿射变换矩阵
- w：图像的宽度
- h：图像的高度
- data：指向图像像素数据的指针
- transparent：透明色
- depth：图像的颜色深度
- palette：调色板指针

返回值:

- null

pushImageAffineWithAA

函数原型1:

```
void pushImageAffineWithAA(const float matrix[6], int32_t w, int32_t h, const T* data)
```

函数原型2:

```
void pushImageAffineWithAA(const float matrix[6], int32_t w, int32_t h, const T1* data, const T2& trans
```

函数原型3:

```
void pushImageAffineWithAA(const float matrix[6], int32_t w, int32_t h, const void* data, color_depth_t
```

函数原型4：

```
void pushImageAffineWithAA(const float matrix[6], int32_t w, int32_t h, const void* data, uint32_t tran
```

功能说明：

- 使用抗锯齿的仿射变换将图像推送到显示屏上

传入参数：

- matrix：6 个浮点数的数组，表示仿射变换矩阵
- w：图像的宽度
- h：图像的高度
- data：指向图像像素数据的指针
- transparent：透明色
- depth：图像的颜色深度
- palette：调色板指针

返回值：

- null

pushImageRotateZoom

函数原型1：

```
void pushImageRotateZoom(float dst_x, float dst_y, float src_x, float src_y, float angle, float zoom_x,
```

函数原型2：

```
void pushImageRotateZoom(float dst_x, float dst_y, float src_x, float src_y, float angle, float zoom_x,
```

函数原型3：

```
void pushImageRotateZoom(float dst_x, float dst_y, float src_x, float src_y, float angle, float zoom_x,
```

函数原型4：

```
void pushImageRotateZoom(float dst_x, float dst_y, float src_x, float src_y, float angle, float zoom_x,
```

功能说明：

- 将图像旋转和缩放后推送到显示屏上

传入参数：

- dst_x：目标图像左上角的 x 坐标
- dst_y：目标图像左上角的 y 坐标
- src_x：源图像左上角的 x 坐标
- src_y：源图像左上角的 y 坐标
- angle：旋转角度（以弧度为单位）
- zoom_x：x 方向的缩放比例
- zoom_y：y 方向的缩放比例
- w：图像的宽度
- h：图像的高度

- data：指向图像像素数据的指针
- transparent：透明色
- depth：图像的颜色深度
- palette：调色板指针

返回值：

- null

pushImageRotateZoomWithAA

函数原型1：

```
void pushImageRotateZoomWithAA(float dst_x, float dst_y, float src_x, float src_y, float angle, float z
```

函数原型2：

```
void pushImageRotateZoomWithAA(float dst_x, float dst_y, float src_x, float src_y, float angle, float z
```

函数原型3：

```
void pushImageRotateZoomWithAA(float dst_x, float dst_y, float src_x, float src_y, float angle, float z
```

函数原型4：

```
void pushImageRotateZoomWithAA(float dst_x, float dst_y, float src_x, float src_y, float angle, float z
```

功能说明：

- 使用抗锯齿的方式将图像旋转和缩放后推送到显示屏上

传入参数：

- dst_x：目标图像左上角的 x 坐标
- dst_y：目标图像左上角的 y 坐标
- src_x：源图像左上角的 x 坐标
- src_y：源图像左上角的 y 坐标
- angle：旋转角度（以弧度为单位）
- zoom_x：x 方向的缩放比例
- zoom_y：y 方向的缩放比例
- w：图像的宽度
- h：图像的高度
- data：指向图像像素数据的指针
- transparent：透明色
- depth：图像的颜色深度
- palette：调色板指针

返回值：

- null

图像颜色转换

setSwapBytes

函数原型:

```
void setSwapBytes(bool swap)
```

功能说明:

- 设置字节交换状态

传入参数:

- swap：字节交换状态
  - true：字节交换
  - false：字节不交换

返回值:

- null

getSwapBytes

函数原型:

```
bool getSwapBytes(void)
```

功能说明:

- 获取由 setSwapBytes 设置的字节交换状态

传入参数:

- null

返回值:

- bool：字节交换状态
  - true：已交换
  - false：未交换

swap565

函数原型:

```
uint16_t swap565( uint8_t r, uint8_t g, uint8_t b)
```

功能说明:

- 根据 R,G,B 分量生成颜色编码

传入参数:

- r: 红色分量，0-255
- g: 绿色分量，0-255
- b: 蓝色分量，0-255

返回值:

- uint16_t：RGB565 颜色编码

swap888

函数原型:

```
uint16_t swap888( uint8_t r, uint8_t g, uint8_t b)
```

功能说明:

- 根据 R,G,B 分量生成颜色编码

传入参数:

- r: 红色分量，0-255
- g: 绿色分量，0-255
- b: 蓝色分量，0-255

返回值:

- uint32_t：RGB888 颜色编码

案例程序:

cpp

```cpp
#include <Arduino.h>
#include <M5GFX.h>
#include <M5Unified.h>

M5GFX display;
uint16_t PngData[1600];
static float Affine_mat[9] = {1, 0, 0,
                              0, 1, 60,
                              0, 0, 1  };

void setup() {
    display.begin();
    display.setColorDepth(24);
    display.setRotation(1);
    display.clear(TFT_WHITE);
    display.setTextFont(&fonts::FreeSansOblique12pt7b);
    display.setTextColor(0xF81F);
    delay(1000);
    uint16_t x = display.width() / 2;
    uint16_t y = display.height() / 2;
    // display.setSwapBytes(true);//If you want to directly assign a 16-bit color code in the lower 
    for (int i = 0; i < 1600; ++i) PngData[i] = display.swap565( 255, 0, 0);//After swapping, it bec

    display.drawCenterString("Push Image Test", x, 10);
    delay(2000);
    display.pushImage(0, y, 320, 5, PngData);
    // display.pushImageDMA(0, y, 320, 5, PngData);
    // Shift 60 units from the origin(0,0) using the affine matrix
    // display.pushImageAffine(Affine_mat, 320, 5, PngData, 80);
    display.pushImageAffineWithAA(Affine_mat, 320, 5, PngData, 40);
    // Rotate and Zoom, by comparing on the screen, you can see the effect of anti-aliasing.
    display.pushImageRotateZoom(x, y, 0, 0, 37, 2, 2, 320, 5, PngData);
    display.pushImageRotateZoomWithAA(x, y, 0, 4, 143, 2, 2, 320, 5, PngData);
}

void loop() {
}
```