Image Display

Introduction to the results of routine experiments

The example code is located in: [Source code/02.Basic/15.Image.py]

We use CanMV IDE to open the sample code and connect K230 to the computer with a USB cable.

This tutorial requires a PNG image.

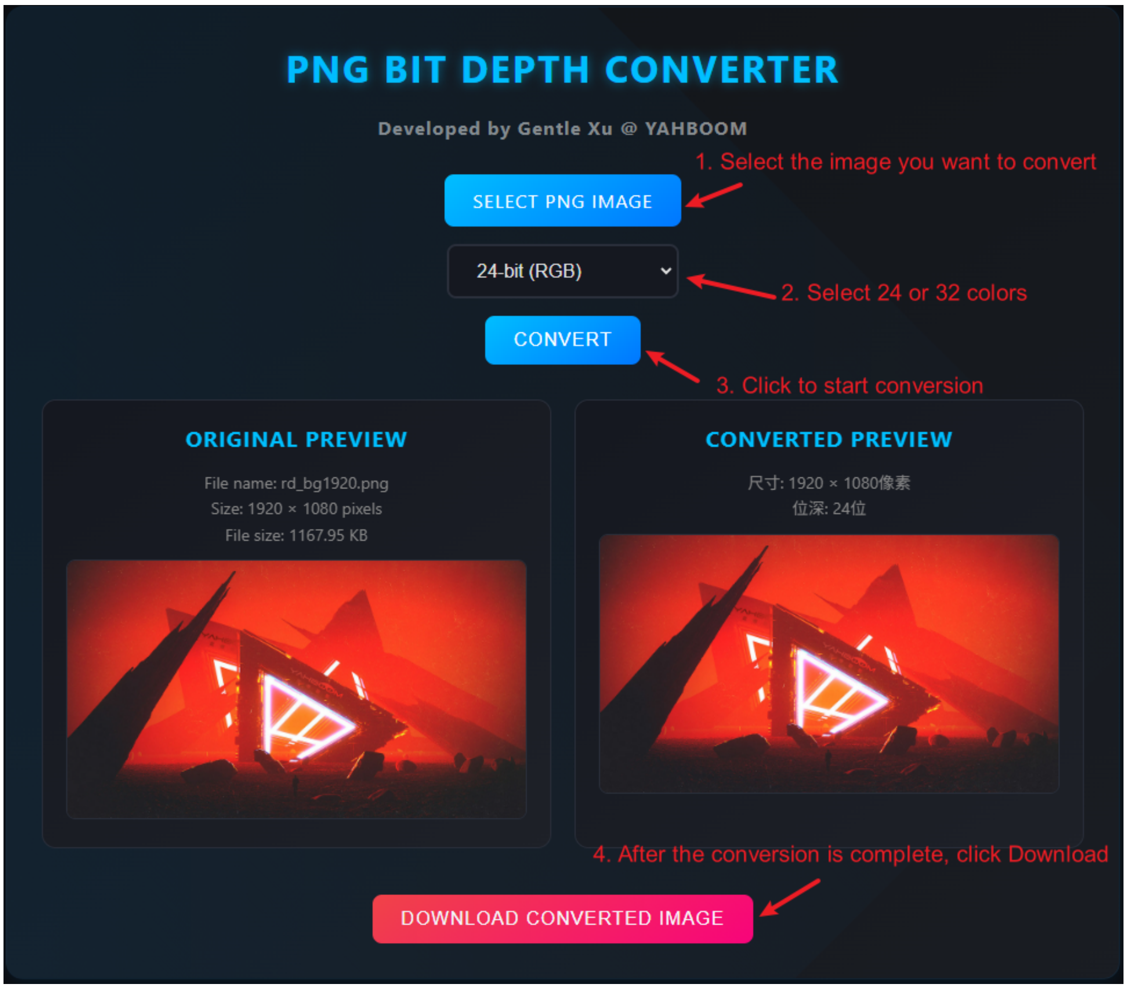
According to the manual, JPEG images are also acceptable, but actual testing shows that the effect is not as good as PNG images, so we can convert the images we want to display to PNG first.

Here I have prepared a picture, the path is as follows [/sdcard/resources/wp.png]

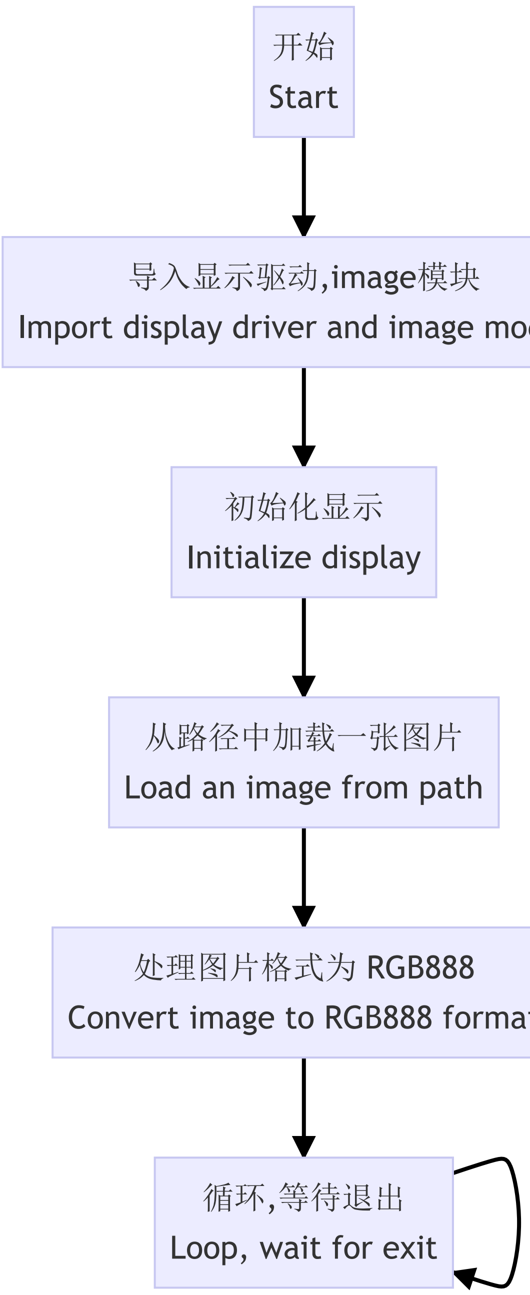
We open the example code and click the run button in the lower left corner. We can see that the image we prepared will be displayed on the screen (and in the frame buffer).

*Note: The image size cannot exceed the size limit of the display device. If using a matching screen, the resolution must be limited to 640x480

Routine code

> *PNG images with a bit depth of 8 cannot be loaded normally, and the IDE will report an error: RuntimeError: given output image colortype or bitdepth not supported for color conversion
>
> You can use the PNG bit depth converter provided in our document to convert the image bit depth to 24 or 32
>
> 
> PNG BIT DEPTH CONVERTER
>
> Developed by Gentle Xu @ YAHBOOM
>
> 1. Select the image you want to convert
>
> 2. Select 24 or 32 colors
>
> 3. Click to start conversion
>
> ORIGINAL PREVIEW
>
> File name: rd_bg1920.png
> Size: 1920 × 1080 pixels
> File size: 1167.95 KB
>
> CONVERTED PREVIEW
>
> 尺寸: 1920 × 1080像素
> 位深: 24位
>
> 4. After the conversion is complete, click Download
>
>

The process of displaying pictures:



The code to display the image is as follows:

The examples in this section require the use of Media's display and media modules. Subsequent multimedia-related operations require the introduction of these two modules.

In addition, the image module is introduced to process image information

```
from  media . display  import  *
from  media . media  import  *
import  image , time

Display . init ( Display . ST7701 , width = 640 , height = 480 , osd_num = 1 ,
to_ide = True )

# If the screen is not selected, use this initialization code:
# Display.init(Display.VIRT, width = 1920, height = 1080)

img = image . Image ( "/sdcard/resources/wp.png" , copy_to_fb = True )
```

```
MediaManager.init ( )

# Here you must convert the png image to rgb888 or rgb565, otherwise it cannot be
displayed
img = img.to_rgb888 ( )

Display . show_image ( img )
while  True :
    pass
```

Image Module

The Image class is a basic object in machine vision processing. This class supports creating image objects from memory areas such as Micropython GC, MMZ, system heap, VB area, etc. In addition, images can be created directly by referencing external memory (ALLOC_REF). Unused image objects will be automatically released during garbage collection, or memory can be released manually.

The supported image formats are as follows:

- BINARY
- GRAYSCALE
- RGB565
- BAYER
- YUV422
- JPEG
- PNG
- ARGB8888 (New)
- RGB888 (New)
- RGBP888 (New)
- YUV420 (new)

Supported memory allocation areas:

- **ALLOC_MPGC** : Micropython managed memory
- **ALLOC_HEAP** : System heap memory
- **ALLOC_MMZ** : multimedia memory
- **ALLOC_VB** : Video buffer
- **ALLOC_REF** : Use the memory of the referenced object without allocating new memory

Constructor

```
image . Image ( path , alloc = ALLOC_MMZ , cache = True , phyaddr = 0 , virtaddr
= 0 , poolid = 0 , data = None )
```

Create an image object from a file path `path`, supporting BMP, PGM, PPM, JPG, and JPEG formats.

```
image . Image ( w , h , format , alloc = ALLOC_MMZ , cache = True , phyaddr = 0
, virtaddr = 0 , poolid = 0 , data = None )
```

Creates an image object of the specified size and format.

- **w** : image width
- **h** : image height
- **format** : image format
- **alloc** : memory allocation method (default ALLOC_MMZ)
- **cache** : Whether to enable memory cache (enabled by default)
- **phyaddr** : physical memory address, only applicable to VB area
- **virtaddr** : virtual memory address, only applicable to VB area
- **poolid** : Pool ID of the VB region, only applicable to the VB region
- **data** : reference to an external data object (optional)

Example:

```
# Create a 640x480 image in ARGB8888 format in the MMZ region
img = image . Image ( 640 , 480 , image . ARGB8888 )

# Create a 640x480 image in YUV420 format in the VB area
img = image . Image ( 640 , 480 , image . YUV420 , alloc = image . ALLOC_VB ,
phyaddr = xxx , virtaddr = xxx , poolid = xxx )

# Create a 640x480 image in RGB888 format using an external reference
img = image . Image ( 640 , 480 , image . RGB888 , alloc = image . ALLOC_REF ,
data = buffer_obj )
```

Manually release image memory:

```
del  img
gc . collect ()
```

For more information about the Image module, please refer to Canaan's official API manual [3.11 Image Processing API Manual — CanMV K230]