

第8讲 | 如何区分图形和图像？

2018-06-12 蔡能

从0开始学游戏开发

[进入课程 >](#)



讲述：蔡能

时长 07:22 大小 3.40M




据我所知，很多人可能都分不清图形和图像这两个概念：一种情况是你可能会觉得区分图形和图像这两个概念并没有什么实质的用途，于是就没关心过；另一种情况是，你可能朦胧中对图形和图像的区别有一些了解，但是不够深入或者不够透彻，也说不出一个所以然。没关系，今天我就来深入浅出地给你讲一下，图形和图像背后的那些事儿。

既然我们是做游戏开发，那首先我们得知道，从专业地角度讲，区分图形和图像对我们的开发工作有什么帮助。简单地说，**搞清楚了游戏开发中绘制、载入、保存的究竟是图形还是图像，你会对接口函数的使用有一个更深入的认识。**

比如，如果是图形接口，可能它的接口函数是：

```
1 Surface* DrawSomething(int start_x, int start_y, int finish_x, int finish_y);
```

如果是图像接口，它的接口函数函数看起来可能是这个样子：

 复制代码

```
1 Surface* LoadFromFile(const string& filename);
```

如何区分图形和图像？

从广义上说，所有我们人肉眼能看到的对象，都是图形。从狭义上说，图形是我们所看到的一种点、线、面的描述对象。

图像，是由数据组成的任意像素点的描述对象。比如我们所看到的照片。在电脑中，图形的显示过程是有一定顺序（比如从左往右）的，而图像则是按照像素点进行显示的。电脑对于图形的编辑、修改更为简单方便，因为单一的图形具有特殊的属性（比如圆圈的直径、颜色等等，因为这些在这个图形建立的时候就固定了下来）。

对于图像进行编辑则非常困难，软件需要用一些特殊的算法来计算图像的色块、区域、描边等等，来安排图像该如何进行编辑，有一些甚至还需要用到深度学习的方法来辨别图像的显示区域、显示的内容等等，所以图像的修改比之图形的修改要困难。

那么你可能就会问了，既然前面说，任何眼睛看到的对象，都是图形，那么我觉得图形也是一种图像，这么说对不对呢？如果按照载体来说，图形也是一种图像，这种说法是对的。因为一张 JPG 图片可能存储的是一幅照片，也可能存储一幅三角形的图形。虽然本质不一样，但是由于存储的形式是以图像的形式存储的，在电脑看来，这个三角形就是一幅图像。但是如果你在游戏中使用函数画出了一个三角形，那就是图形了。

所以，严格来说，**图形其实是图像的一种抽象表现形式**。一般来讲，图形的轮廓并不复杂，比如一个圆圈、一个方块、一个三角形、一条线、某些几何图形、工程上面使用的图纸和 CAD 等，这些都属于图形。图形的色彩也并不是很丰富。而图像一般都有复杂的轮廓、非常多的细节和颜色（当然也有纯单一的颜色，比如黑白照片）。




所以，准确地说，图形和图像有不同的模式。当然，从计算机最底层的程序（显卡处理）来看，绘制图形和图像所经过的硬件处理几乎是一样的。一般显卡会经过这些流程进行图形、图像计算（2D）、显存，用来存取图形图像内容，GPU 计算图像图像内容并渲染，最后输出到显示器。

从**图像的呈现方式**讲，只有通过图像的方式去呈现“图形”这个对象，才能看到图形，而在开发游戏的过程中，图形和图像的编程方式是截然不同的。比如我们要画线，那么可能会使用到一个叫 DrawLine 的函数。该函数里面需要输入线条的起始坐标，这就是图形的绘制方式。而在接下来的过程中，我将教你如何绘制图形和图像，并呈现出来。

跟我一起绘制图形和图形

现在，我们先用 Pygame 游戏库来建立一个窗体，然后开始绘制图形、载入图像。

在第五节的时候，我们已经讲过 Pygame 的安装和配置。在第六节的时候，我们讲过如何建立一个 Windows 窗体。现在从上到下，我们一起看一下这段代码。

 复制代码


```
1 import pygame
2 pygame.init()
3 caption=pygame.display.set_caption('Python App')
4 screen=pygame.display.set_mode([320,200]) # 窗口大小为 320*200
5 while True:
```

```
6     for event in pygame.event.get():
7         if event.type == pygame.QUIT:
8             pygame.quit()
9     pygame.display.update()
10    screen.fill([255,255,255]) # 用白色填充窗体
11 sys.exit()
```

在这段代码中，首先，我们需要告诉 Python 我们要引入 Pygame。然后 Pygame 进行初始化 (init)。在这个初始化的函数里，Pygame 会初始化屏幕、声音、事件、按钮等一系列需要初始化的东西。随后，我们利用 Pygame 的 display 对象的 set_caption 函数来设置窗体的文字，将这个设置后的对象返回给 caption 变量。随后，再使用 set_mode 函数设置窗口大小，将窗口大小设置为 320x200 分辨率，将返回对象赋值给 screen 变量，最后 screen 拿到窗口句柄后，使用 fill 函数设置填充窗体的颜色，在这里填充的颜色是白色。

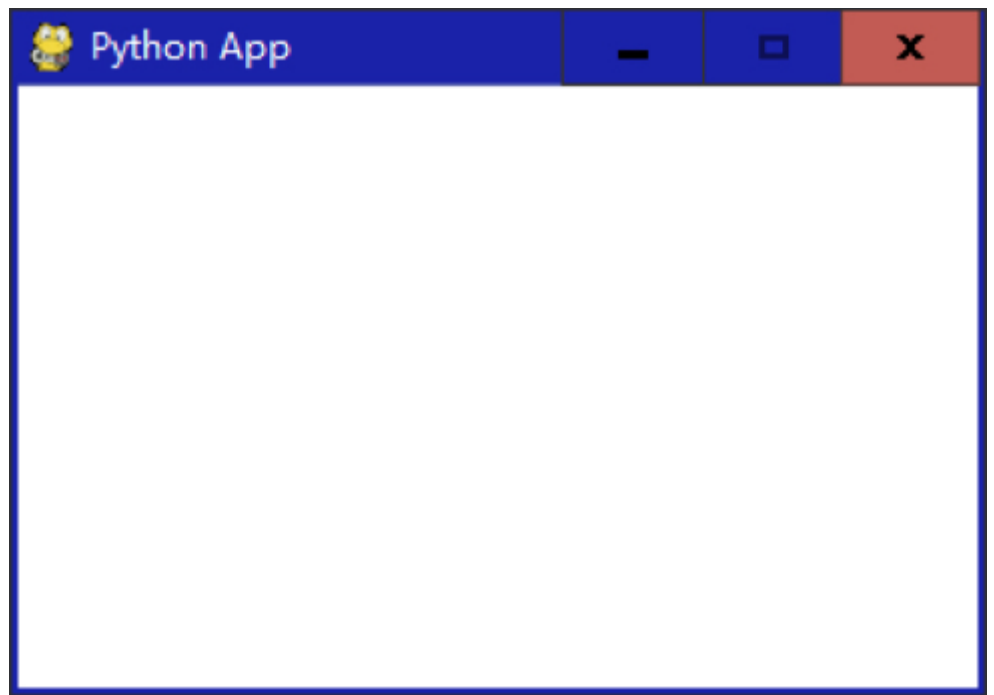
我们可以看到，使用 Pygame 游戏库来建立一个 Windows 窗体比前面我们提到的任何一种方式都快。那是因为**Pygame 封装了建立窗体的代码和图形显示模块**。

我们在前面提到，**一个游戏是在一个大循环下形成的**，所以这里我们要补上一个大循环以确保这个程序不会立刻退出。


 复制代码

```
1 while True:
2     for event in pygame.event.get():
3         if event.type == pygame.QUIT:
4             pygame.quit()
5     pygame.display.update()
6     screen.fill([255,255,255]) # 用白色填充窗体
7 sys.exit()
```

这段代码的意思是，当条件为真 (True) 的时候（条件总是为真），进行一个循环。事实上这是个死循环，如果没有下面的退出代码的话。那么在这个循环里，从 Pygame 的 event 事件列表中取出 event 事件，然后进行判断，如果 event 的类型是退出类型（点击右上角的 X 关闭按钮），那么 Pygame 就退出，这个 quit 函数就直接退出 while 大循环了。最终系统也退出 sys.exit。




现在我们要在窗体上放上一个矩形和圆。我们先使用 `rect` 函数来画一个矩形：

 复制代码

```
1 pygame.draw.rect(screen, [255,0,0], [150,10,0,40],0)
```

其中，`draw` 中 `rect` 的定义为：`rect(目标画布, 颜色, 位置, 宽度)`。

我们也可以用类似的方法来画一个圆：

 复制代码

```
1 pygame.draw.circle(screen,[0,0,0],[top,left],20,1)
```

然后我们使用 `pygame.draw.circle()` 用来画圆形。`circle` 函数具有 5 个参数：

目标画布，在这里是 `screen`


颜色

由左侧点和顶部点组成的圆形初始位置

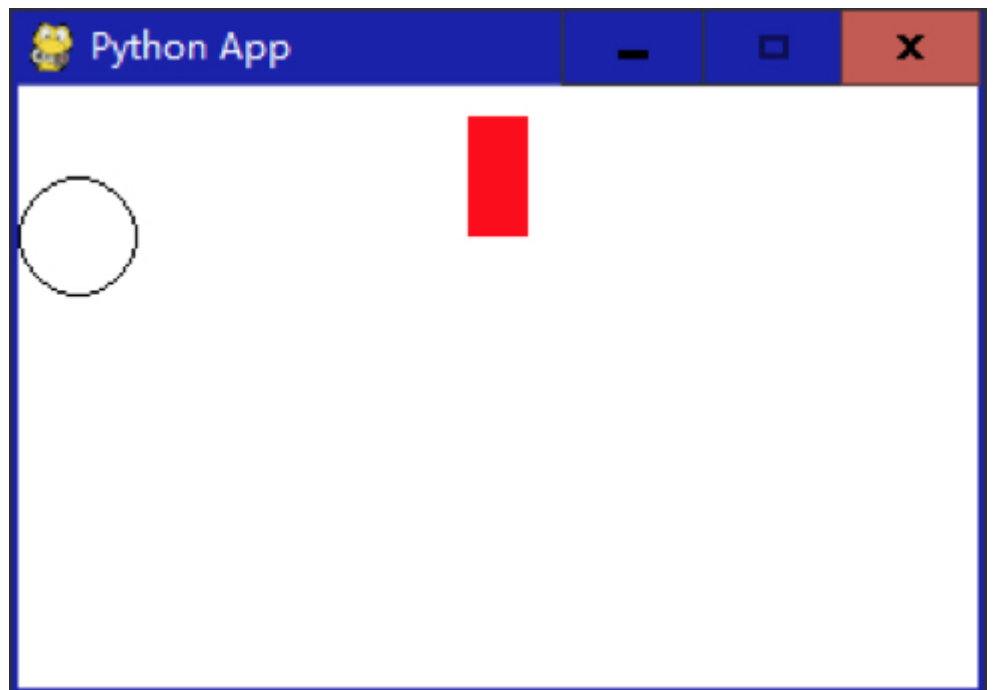
直径

宽度

现在我们将所有的代码合并起来看一下：

 复制代码

```
1 import pygame
2 pygame.init()
3 caption=pygame.display.set_caption('Python App')
4 screen=pygame.display.set_mode([320,200]) # 窗口大小为 640*480
5 while True:
6     for event in pygame.event.get():
7         if event.type==pygame.QUIT:
8             pygame.quit()
9     pygame.draw.rect(screen,[255,0,0],[150,10,20,40],0)
10    pygame.draw.circle(screen,[0,0,0],[20,50],20,1)
11    pygame.display.update()
12    screen.fill([255,255,255])# 用白色填充窗口
13    sys.exit()
```



所以我们很容易就能看出来，**在 Pygame 游戏开发库里面，画一个图形是很容易的事情，你不需要知道太多的细节，只要将位置和颜色或者内容填充进去就可以了。**

我们可以在 Pygame 中使用 `Pygame.image.load` 来加载图像文件，这个函数支持各种图片格式。我们使用这个方法加载一副 PNG 图片：

```
1 obj = pygame.image.load("test.png").convert_alpha()
```

使用 `convert_alpha` 函数是因为这个函数会使用透明方法来绘制，所以我们在加载一个拥有 alpha 通道的图片的时候（比如 TGA、PNG）的时候，可以使用这个方式。

然后使用 `blit` 方法将图像绘制出来：

```
1 screen.blit(obj, (20,10))
```

或许你会问，`blit` 是什么函数，我在这里简单介绍一下，`blit` 这个函数会以各种函数形式出现在图形引擎的函数里面，比如 `FastBlit` 等等。这个函数具体负责将图像从某一个平面复制到另一个平面，或者将图像从内存复制到屏幕。简而言之，这个函数的功能就是将图像“绘制”在游戏窗体的屏幕上。

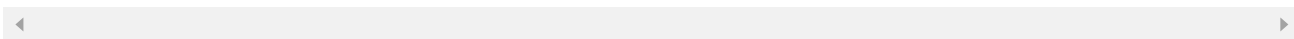
现在继续来看看 `blit` 函数。`blit` 函数的第一个参数是加载完成的返回对象，第二个参数是绘制的坐标位置。最后我们需要 `update`（更新）整个游戏窗体的绘制内容。

我们把载入图像的代码整合到刚才的代码中一块儿看一下。

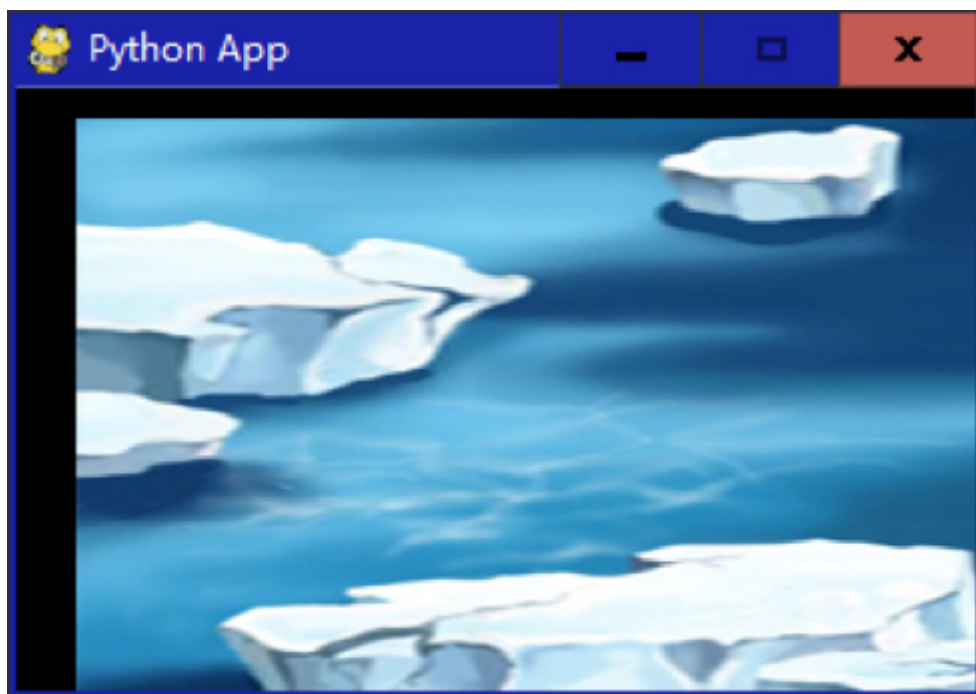
```
1 import pygame
2 pygame.init()
3 caption=pygame.display.set_caption('Python App')
4 screen=pygame.display.set_mode([320,200]) # 窗口大小为 640*480
5
6 obj = pygame.image.load("test.png").convert_alpha()
7
8 while True:
9
10     for event in pygame.event.get():
11         if event.type==pygame.QUIT:
12             pygame.quit()
13             sys.exit()
14     screen.blit(obj, (20,10))
15     pygame.display.update()
```



```
16 screen.fill([255,255,255])# 用白色填充窗口
```



最后呈现的效果是这样的：



小结

这一节，我带你学习了图形和图像的区别，使用 Pygame 绘制了最基础的图形，最后我们通过代码载入一副 PNG 图像并在屏幕上绘制出来。

给你留一个小练习吧。

请你结合上述代码，在游戏执行的大循环内，在游戏的窗体里面，绘制出一个从左到右移动的矩形、圆形或者图像。

之后，针对一些实操性强的内容，我都会适时给你留一些必要的练习。希望你每次都能动手去练习一下。同时，也欢迎你留言，说出你在练习中的疑惑和成果。温故而知新，相信你会有更多的收获！

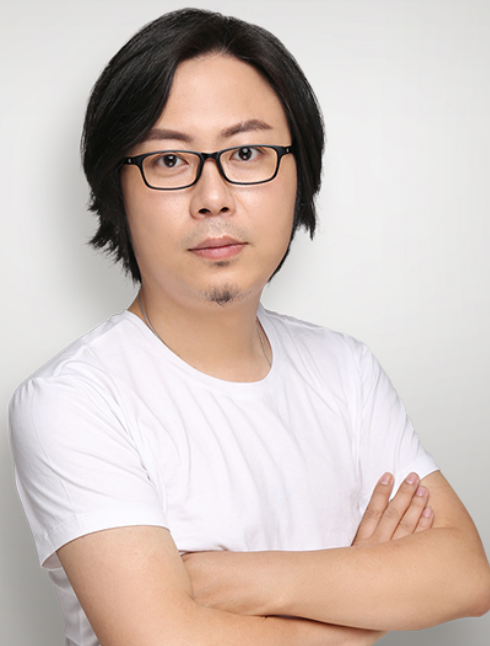
我在下一节的挑战中等你！

从0开始学游戏开发

你的游戏开发入门第一课

蔡能

原网易游戏引擎架构师
资深游戏底层技术专家



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 第7讲 | 如何建立一个Windows窗体？

下一篇 第9讲 | 如何绘制游戏背景？

精选留言 (5)

写留言



茂陵刘郎秋...

2018-06-12

4

不知道是排版问题还是什么？画图那一块代码照着敲出来无法实现，把画图代码放到while循环里就可以运行了



王鲜申

2018-12-01

可以在每次画图之前加一个 for 循环，然后用一个变量控制图形的位置，用 for 循环中的终止条件控制移动速度



奇小易

2018-07-13



试了一下画圆从左到右，就是调整圆的横坐标，但是直接运行就跑得太快了。而就想使用time.sleep函数加降低刷新频率，结果是屏幕黑的，显示未响应。



以往

2018-06-12



不同的情景下，游戏人物的动作缓急都各有不同。一般是倾向于改变帧速，还是在同一帧速下使用调整了动作幅度的图片来达到效果？

作者回复: 调整幅度也是一种方案，多线程里绘制需要的内容，然后改变帧率也是一种方案



呵呵

2018-06-12



后面的课程准备用pygame开发客户端，c++开发server？

展开 ∨

作者回复: 会带一部分c，python也有

