pygame窗口

窗口的初始化其实是在

```
pygame.init()
```

的时候就已经完成了的。在我们使用

```
screen = pygame.display.set_mode(size=(0,0), flag=0, depth=0, vsync=0)
```

的时候，就获得了一个名字叫screen的窗口（实质上是一个surface对象），各项参数代表如下：

> - size代表窗口的宽和高
> - flag代表窗口的类型，下文会细说。
> - depth表示色深，建议不要自行设置。
> - vsync表示是否开启**垂直同步**，可以不进行设置。

所以我们需要设置的基本上只有size和flag。

size这里就不多解释了，除了代表窗口的宽和高之外，有兴趣的同学可以试试把宽和高都设置为0，看看会发生什么

flag表示窗口的类型，常用的有以下几种：

类型名	含义
pygame.FULLSCREEN	全屏
pygame.RESIZEABLE	可以通过拖曳调整窗口大小
pygame.NOFRAME	无边框
pygame.SCALED	自适应缩放
pygame.SHOWN	默认
pygame.HIDDEN	后台启动，老实说我不太明白它存在的意义

需要说明的是，这几个是可以通过操作符 | 来叠加的，比如既是RESIZEABLE又是SCALED

```
flag = pygame.RESIZEABLE | pygame.SCALED
```

原文解释如下，感兴趣的同学可以自行学习：

> - pygame.display.**set_mode**()
>
>   *Initialize a window or screen for display*
>
>   set_mode(size=(0, 0), flags=0, depth=0, display=0, vsync=0) -> Surface
>
>   This function will create a display Surface. The arguments passed in are requests for a display type. The actual created display will be the best possible match supported by the system.

Note that calling this function implicitly initializes `pygame.display`, if it was not initialized before.

The size argument is a pair of numbers representing the width and height. The flags argument is a collection of additional options. The depth argument represents the number of bits to use for color.

The Surface that gets returned can be drawn to like a regular Surface but changes will eventually be seen on the monitor.

If no size is passed or is set to `(0, 0)` and pygame uses `SDL` version 1.2.10 or above, the created Surface will have the same size as the current screen resolution. If only the width or height are set to `0`, the Surface will have the same width or height as the screen resolution. Using a `SDL` version prior to 1.2.10 will raise an exception.

It is usually best to not pass the depth argument. It will default to the best and fastest color depth for the system. If your game requires a specific color format you can control the depth with this argument. Pygame will emulate an unavailable color depth which can be slow.

When requesting fullscreen display modes, sometimes an exact match for the requested size cannot be made. In these situations pygame will select the closest compatible match. The returned surface will still always match the requested size.

On high resolution displays(4k, 1080p) and tiny graphics games (640x480) show up very small so that they are unplayable. SCALED scales up the window for you. The game thinks it's a 640x480 window, but really it can be bigger. Mouse events are scaled for you, so your game doesn't need to do it. Note that SCALED is considered an experimental API and may change in future releases.

The flags argument controls which type of display you want. There are several to choose from, and you can even combine multiple types using the bitwise or operator, (the pipe "|" character). Here are the display flags you will want to choose from:

```
pygame.FULLSCREEN    create a fullscreen display
pygame.DOUBLEBUF     (obsolete in pygame 2) recommended for HWSURFACE
or OPENGL
pygame.HWSURFACE     (obsolete in pygame 2) hardware accelerated, only
in FULLSCREEN
pygame.OPENGL        create an OpenGL-renderable display
pygame.RESIZABLE     display window should be sizeable
pygame.NOFRAME       display window will have no border or controls
pygame.SCALED        resolution depends on desktop size and scale
graphics
pygame.SHOWN         window is opened in visible mode (default)
pygame.HIDDEN        window is opened in hidden mode
```

New in pygame 2.0.0: `SCALED`, `SHOWN` and `HIDDEN`

By setting the `vsync` parameter to `1`, it is possible to get a display with vertical sync, but you are not guaranteed to get one. The request only works at all for calls to `set_mode()` with the `pygame.OPENGL` or `pygame.SCALED` flags set, and is still not guaranteed even with one of those set. What you get depends on the hardware and driver configuration of the system pygame is running on. Here is an example usage of a call to `set_mode()` that may give you a display with vsync:

> ```
> flags = pygame.OPENGL | pygame.FULLSCREEN
> window_surface = pygame.display.set_mode((1920, 1080), flags, vsync=1)
> ```
>
> Vsync behaviour is considered experimental, and may change in future releases.
>
> *New in pygame 2.0.0:* `vsync`
>
> Basic example:
>
> ```
> # Open a window on the screen
> screen_width=700
> screen_height=400
> screen=pygame.display.set_mode([screen_width, screen_height])
> ```
>
> The display index `0` means the default display is used. If no display index argument is provided, the default display can be overridden with an environment variable.

那么下面考虑当设置为RESIZEABLE时的情况：

如果我们的游戏是依赖于窗口大小的（比如说边界就是在窗口边缘），那么当我们设置为RESIZEABLE时，如果我们改变游戏窗口的尺寸，那么我们之前设置的固定的窗口大小就不再适用于新的窗口大小。所以我们可以通过：

```
windowsInfo = pygame.display.Info()
```

来获得窗口大小。它返回的是一个对象，拥有很多属性，这里不再多说，只有两个我们用的到的

- current_h,代表现在窗口的高
- current_w,代表现在窗口的宽

其它的原文如下，感兴趣的可以自行学习：

> - pygame.display.**Info**()
>
>   *Create a video display information object*
>
>   Info() -> VideoInfo
>
>   Creates a simple object containing several attributes to describe the current graphics environment. If this is called before `pygame.display.set_mode()` some platforms can provide information about the default display mode. This can also be called after setting the display mode to verify specific display options were satisfied. The VidInfo object has several attributes:
>
>   ```
>   hw:         1 if the display is hardware accelerated
>   wm:         1 if windowed display modes can be used
>   video_mem:  The megabytes of video memory on the display. This is 0 if
>               unknown
>   bitsize:    Number of bits used to store each pixel
>   bytesize:   Number of bytes used to store each pixel
>   masks:      Four values used to pack RGBA values into pixels
>   shifts:     Four values used to pack RGBA values into pixels
>   losses:     Four values used to pack RGBA values into pixels
>   blit_hw:    1 if hardware Surface blitting is accelerated
>   blit_hw_CC: 1 if hardware Surface colorkey blitting is accelerated
>   blit_hw_A:  1 if hardware Surface pixel alpha blitting is accelerated
>   blit_sw:    1 if software Surface blitting is accelerated
>   ```

```
blit_sw_CC: 1 if software Surface colorkey blitting is accelerated
blit_sw_A:  1 if software Surface pixel alpha blitting is accelerated
current_h, current_w:  Height and width of the current video mode, or
            of the desktop mode if called before the display.set_mode
            is called. (current_h, current_w are available since
            SDL 1.2.10, and pygame 1.8.0). They are -1 on error, or if
            an old SDL is being used.
```

其它常见的还有例如

- screen.flip() 用于更新窗口
- screen.set_caption(Caption) 用于设置窗口标题
- screen.set_icon(path) 用于设置窗口的图标，建议尺寸32x32