python™

PYGAME

briup

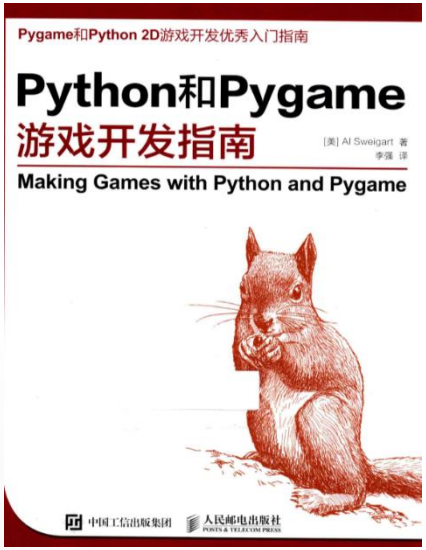
Pygame

- pygame是一个利用SDL库写游戏库，是一款成熟、简洁易用的2D游戏开发库
- SDL（Simple DirectMedia Layer）

  是一套开源的跨平台多媒体开发库，C语言库

Pygame

- 学习资料：
- 官网 https://www.pygame.org/news
- 《python和pygame游戏开发指南》
- 中文手册
- 目光博客



Pygame

- 模块（34个）主要模块：

pygame.display	访问显示设备
pygame.surface	管理图像和屏幕
pygame.image	管理图片
pygame.draw	绘制形状、线和点
pygame.rect	管理矩形区域
pygame.mouse、pygame.key	管理鼠标、按键
Pygame.miuse、pygame.font	音乐、字体
pygame.event	管理事件
pygame.sprite	管理移动图像

Display模块

- Display是用于控制窗口和屏幕显示的模块

display.init	初始化显示模块
display.quit	关闭显示界面
display.set_mode	创建一个显示窗口
display.flip	刷新整个屏幕
display.update	局部属性屏幕
display.set_icon	设置图标
display.set_caption	设置标题

Display模块

- init（）

在使用display模块时，必须先进行初始化，display模块将选择后台的显卡驱动

- set_mode(resolution=(0,0), flags=0, depth=0)

resolution：宽，高

flags:　选项

FULLSCREEN　创建一个全屏窗口

RESIZABLE　创建一个可以改变大小的窗口

NOFRAME　创建一个没有边框的窗口

depth: 颜色深度

返回surface实例

Display模块

- Flags 是 pygame 定义的常量，使用时，要导入

  from pygame.locals import *

- Depth=0，pygame 将自动选择最适合的颜色深度，修改会进行重新渲染，不建议设置

- 返回值为 surface的一个实例

Surface 模块

- Surface 类 用于表示 pygame中 的图像

Surface((width, height), flags=0, depth=0, masks=None)

Surface((width, height), flags=0, Surface)

display.set_mode( )

image.load( )

...

Surface 模块

- Image模块用于加载和保存图片

  支持加载一下格式：

  JPG、PNG、GIF (non-animated)、BMP、PCX、TGA (uncompressed)、TIF、LBM (and PBM)、PBM (and PGM, PPM)、XPM

  支持保存：

  BMP、TGA、PNG、JPEG

- pygame.image.load(filename) -> surface
- pygame.image.save (Surface, filename)

Surface 模块

- Surface类的常用方法：

1 surface.convert( )

surface.convert_alpha( )

这两个函数常用来优化像素格式，是pygame可以更加迅速的绘制图片。除此之外，还可以定制像素格式，修改颜色深度等

2 绘制（blit 位块传送）

surface.blit(source, dest, area=None, special_flags = 0)

source: surface类型参数，将制定的surface对象绘制到当前surface对象上

dest：指定绘制的位置（x,y）

area：指定绘制图像上的某一区域 （x,y,h,w）或 rect 对象,None为整个对象

函数返回 rect对象，实际绘制的矩形区域

Surface 模块

3 pygame.Surface.lock（）

pygame.Surface.unlock（）

对surface对象的内存进行加锁和解锁

4 get_size() 获得surface的宽高

gat_rect()获得surface的矩形区域

5 绘制完成，要刷新屏幕显示，否则没有效果

display.flip( )　　刷新整个屏幕

display.update( )　　当传递rect参数时，刷新制定区域

Rect 模块

- Rect 类 用来描述矩形区域

Rect 构造方法：

Rect(left, top, width, height) -> Rect

Rect((left, top), (width, height)) -> Rect

Rect(object) -> Rect

Rect

- Rect对象有几个虚拟属性，可用于移动和对齐Rect：

  x,y

  top, left, bottom, right

  topleft, bottomleft, topright, bottomright

  midtop, midleft, midbottom, midright

  center, centerx, centery

  size, width, height

  w,h

Rect

- Rect类提供了一系列生成、调整、移动、翻转、拷贝、碰撞检测等方法。常用方法如下：

Rect.copy（）	拷贝
Rect.move(x,y)	移动
Rect.move_ip(x,y)	移动，改变调用者
Rect.inflate/infate_ip（x,y）	围绕中心 增大或缩小，可以为负数
Rect.clamp/clamp_ip(rect)	将一个rect移动到另一个rect中心
Rect.fit(rect)	按指定rect的宽高比调整原rect的大小
Rect.clip(rect)	获得两个rect的重叠部分

move()
move()
fit()

Rect

- 碰撞检测

Rect.contains(rect)	是否包含指定区域
Rect.collidepoint(x,y)	是否包含指定点
Rect.colliderect（rect)	两个rect是否重叠
Rect.collidelist(list)	是否与指定列表中的多个rect重叠， 真 返回第一个重叠的rect的下标 假 返回-1
Rect.collidelistall(list)	同上，真 返回rect的列表，夹返回[]
Rect.collidedict(dict)	同上，检测的是字典里键对应的值
Rect.collidedictall(dict)	与collidelistall(list) 类似

Draw 模块

- Draw模块用于在surface上画简单的图形
- pygame.draw.rect (surface,color,Rect,with=0)

color : 颜色

ract ：矩形区域

with ：边框，如果with=0,画实心矩形，with不为0，只画边框

Draw 模块

- Draw模块用于在surface上画简单的图形

pygame.draw.polygon (Surface, color, pointlist, width=0)	画多边形 pointlist 点的列表
pygame.draw.circle (Surface, color, pos, radius, width=0)	画圆 ， pos 圆心 radius半径
pygame.draw.ellipse (Surface, color, Rect, width=0)	画椭圆（矩形画椭圆法） Rect(x,y,w,h)
pygame.draw.arc (Surface, color, Rect, start_angle, stop_angle, width=1)	画弧线 (椭圆法画弧线) start_angle 开始位置 stop_angle 结束位置

Draw 模块

- Draw模块用于在surface上画简单的图形

pygame.draw.line (Surface, color, start_pos, end_pos, width=1)	画线
pygame.draw.lines (Surface, color, closed, pointlist, width=1)	画多条线 closed=True多条线连接
pygame.draw.aaline (Surface, color, startpos, endpos, blend=1)	同line,抗锯齿
pygame.draw.aalines (Surface, color, closed, pointlist, blend=1	同lines,抗锯齿

Mouse 模块

- Pygame中mouse模块用来管理鼠标，提供以下接口：

1 pygame.mouse.get_pressed ( )

获取鼠标按键的情况（是否被按下）

返回 (button1, button2, button3) 代表左键、滚轮、右键

2 pygame.mouse.get_pos ( )

获取鼠标光标的位置 返回（x,y）

3 pygame.mouse.set_pos(x,y)

设置鼠标光标的位置

4 pygame.mouse.set_visible(bool)

隐藏或显示鼠标光标

Key 模块

- pygame.key.get_pressed（）

  获取所有按键状态，按下为true,返回一个元祖

  可以通过按键的索引，查询某一个按键的状态

- pygame.key.name（key）

  获得按键名称

Event 模块

- Pygame中 使用 event 模块处理事件

当事件触发时，pygame会将事件实例化成一个事件对象，放入事件队列中，处理时从该队列中获去事件。

事件对象一般包含type属性，来记录是哪一种事件发生。同时附带一些额外的属性来记录事件产生的数据。

type的值是一个整数，唯一标识一种事件。

事件	产生途径	属性
QUIT	用户按下关闭按钮	none
ATIVEEVENT	鼠标移除移入屏幕	gain, state
KEYDOWN	键盘被按下	unicode, key, mod
KEYUP	键盘被放开	key, mod
MOUSEMOTION	鼠标移动	pos, rel, buttons
MOUSEBUTTONDOWN	鼠标按下	pos, button
MOUSEBUTTONUP	鼠标放开	pos, button
JOYAXISMOTION	游戏手柄(Joystick or pad)移动	joy, axis, value
JOYBALLMOTION	游戏球(Joy ball)移动	joy, axis, value
JOYHATMOTION	游戏手柄(Joystick)移动	joy, axis, value
JOYBUTTONDOWN	游戏手柄按下	joy, button
JOYBUTTONUP	游戏手柄放开	joy, button
VIDEORESIZE	Pygame窗口缩放	size, w, h
VIDEOEXPOSE	Pygame窗口暴露	none
USEREVENT	触发了一个用户事件	code

Event 模块

处理事件要求我们从事件队列中拿到事件，并执行对应操作。

Pygame提供了以下队列操作方法：

- pygame.event.get ( )

  get ( )　　->　Eventlist　获取当前事件队列

  get(type)　->　Eventlist　获取指定类型的事件的列表

  get(typelist) -> Eventlist　获取指定类型列表的事件的列表

注意：队列的最大长度为128，事件获取后，就会从队列中删除

Event 模块

- pygame.event.poll ( ) -> EventType instance

  获取单个事件，如果为空，返回 pygame.NOEVENT

- pygame.event.wait ( ) -> EventType instance

  阻塞等待获取单个事件

- pygame.event.peek( ) -> bool

  检测某个事件是否在队列中

Mouse 事件

鼠标的状态改变，会触发三种事件

- 鼠标按下：MOUSEBUTTONDOWN

  携带数据：pos:(x,y)　button:n

  左键 1 中键2　右键3　向上滑4 向下滑5

- 鼠标弹起：MOUSEBUTTONUP
- 鼠标移动：MOUSEMOTION

  rel：相对上一次位置

  pos:(x,y)

  buttons:(0,0,0)　每个鼠标按下状态

（鼠标移除屏幕，不会触发上面的事件）

按键事件

- KEYUP 按键弹起
- KEYDOWN 按键按下

额外的属性：key 记录了那一个按键

自定义事件

- 1 定义事件的唯一标识（事件类型标识符）

  该整数只能在USEREVENT与NUMEVENT之间，及24到32中间

- 2 触发自定义事件

  2.1 使用Event( ) 函数创建一个事件实例

  Event(type, dict) -> EventType instance

  Event(type, **attributes) -> EventType instance

  传递事件类型、携带数据的字典或关键字参数

自定义事件

2.2 将事件从送事件队列

方法一：pygame.event.post(event)

将参数传递的event放在队列尾部

方法二：pygame.time.set_timer(eventid, milliseconds)

定时发送事件到消息队列

eventid : 事件类型Id

milliseconds : 间隔时间

注意：第一次发送事件是从调用该函数后 millisecends 毫秒后才向队列发送。当毫秒数设置为0时，停止发送。

Time 时间模块

- time模块用来管理时间和帧信息时间以毫秒为单位

  常用函数:

  pygame.time.get_ticks(n)　获得自pygame.init()后的时间

  pygame.time.delay(n)　　延时n毫秒

  pygame.time.set_timer(eventid, milliseconds)

  　　定时发送事件到事件队列，时间=0时，为停止定时

Time 时间模块

- 控制帧率

帧率或者刷新速率是程序每秒绘制图形的数目，用FPS或帧/秒来表示，帧率较低时，游戏看上去会顿卡。

TIME

- 控制帧率

1. pygame.time.Clock ( )

实例化一个Clock对象，来跟踪时间时钟，还提供了一些功能来帮助控制游戏的帧率。

2. 调用tick方法

Clock.tick (framerate=0) -> milliseconds

framerate =0，将返回上一次调用该函数到现在的时间

framerate 不为 0，将产生延时,保证保持游戏的运行速度比给定的节拍每秒要慢。例如，给定tick(40),没秒不会超过40帧

3. pygame.time.Clock.get_fps（）

获得当前帧率

文字

- Font模块用来管理文字的显示
- Pygame中使用文字通常分为以下三步：

  1 加载字库

  2 准备文本

  3 绘制

Font

- 加载字库

  pygame 支持 TTF 的字体文件，TTF（TrueType fonts）

  windows 在C:\Windows\Fonts下

  linux 在/usr/share/fonts/truetype下

- pygame.font.Font 用来创建一个字体对象

  Font(filename, size) -> Font

  filename : 字库文件的路径（文件名）

  size：字体大小

Font

■Font对象拥有以下方法

- set_underline(bool)　　设置下划线
- set_bold(bool)　　设置加粗
- set_italic(bool)　　设置斜体
- size(text)　　检测渲染该文本的宽度和高度
- render(text, antialias, color, background=None)

text:要绘制的文字　　antialias：是否抗锯齿

color:字体的颜色　　background：背景颜色

render函数用来返回一个surface对象，用于绘制文本

声音

- pygame.mixer 是一个用来处理声音的模块，其含义为“混音器”。

- pygame.mixer.init

  init(frequency=22050, size=-16, channels=2, buffer=4096)

  Pygame.init()会默认调用该函数初始化混音器

pygame.mixer.stop（）　停止播放所有声道

pygame.mixer.pause（）　暂时停止播放所有声道

pygame.mixer.unpause ()　恢复暂停播放声道

Sound 声音类

- Sound用来加载声音片段，支持wav、ogg格式的音频

pygame.mixer.Sound类提供以下方法

pygame.mixer.Sound (filename) - > 加载音频，实例化sound对象

pygame.mixer.Sound.play　　播放

pygame.mixer.Sound.stop　　停止

pygame.mixer.Sound.fadeout　　渐渐停止

pygame.mixer.Sound.set_volume　设置音量 值0-1

Music 音乐

- music用来加载流式音频，支持mp3、ogg格式的音频
- mucis 与sound不同，混合器同一时间只能加载一首音乐

pygame.mixer.music.load（） 加载音乐，无返回值

pygame.mixer.music.play（loops） 播放音乐,loops循环次数

pygame.mixer.music.stop（）停止

pygame.mixer.music.pause（）暂停

pygame.mixer.music.unpause（）恢复

pygame.mixer.music.fadeout（）渐渐停止

pygame.mixer.music.set_volume（）设置音量

精灵与精灵组

- 精灵可以认为成是一个个小图片，一种可以在屏幕上移动的图形对象，并且可以与其他图形对象交互。精灵图像可以是使用 pygame 绘制函数绘制的图像，也可以是原来就有的图像文件。精灵，简单来说是一个会动图片或小实体。
- Pygame中封装Sprite类，提供一些描述精灵的属性，以及用来与其他精灵交互的方法。比如碰撞检测等。Sprite类作为游戏中不同类型对象的基类。该基类不直接用于创建实例，而常用于继承。

Sprite类

- pygame.sprite.Sprite　类

Sprite(*groups) -> Sprite

构造一个精灵，并将该精灵添加到参数说指的精灵组中

pygame.sprite.Sprite.add (*groups)	添加该精灵到给出的精灵组中
pygame.sprite.Sprite.remove (*groups)	从这些精灵组中，移除该精灵
pygame.sprite.Sprite.kill （）	从所有精灵组移除该精灵
pygame.sprite.Sprite.alive（）	精灵属于某一个精灵组
pygame.sprite.Sprite.groups（）	返回所有包含该精灵的精灵组
pygame.sprite.Sprite.update（*args）	预留函数，程序员决定更新内容

精灵组

- pygame.sprite.Group　精灵组

Group(*sprites) -> Group　　创建一个包含给定精灵的组

pygame.sprite.Group.sprites()　　返回该组中的所有精灵（列表）

pygame.sprite.Group.copy()　　拷贝一个精灵组

pygame.sprite.Group.add(*sprites) 添加一个或多个精灵

pygame.sprite.Group.remove(*sprites)　删除一个或多个精灵组

pygame.sprite.Group.has (*sprites)　　判断组中是否包含所有的精灵

pygame.sprite.Group.empty（）　　删除该小组里的所有精灵

精灵组

- pygame.sprite.Group　精灵组

pygame.sprite.Group.update(*args)

调用组中每个精灵的update函数，对应参数也将传递给精灵的update中

pygame.sprite.Group.draw(Surface)

绘制每个精灵，要求每个精灵有image属性，来加载绘制的图片

pygame.sprite.Group.clear(Surface, background)

擦除上一次draw绘制的精灵，用给出的背景填充对应的位置

精灵组

sprite模块的其他方法：精灵与精灵碰撞检测

- pygame.sprite.collide_rect(sleft, sright) -> bool

  检测两个精灵碰撞，需要精灵的rect属性

- pygame.sprite.collide_mask(SpriteLeft, SpriteRight) -> point

  返回碰撞的坐标

  掩码比较：将原图转为为位图，按位相与，比较的速度较快

  需要有mask属性来保存该精灵的位图

  生成位图的方法

  sprite.mask = pygame.mask.from_surface(sprite.image)

精灵组

sprite模块的其他方法：精灵与精灵碰撞检测

- pygame.sprite.collide_rect(sleft, sright) -> bool

  检测两个精灵碰撞，需要精灵的rect属性

- pygame.sprite.collide_mask(SpriteLeft, SpriteRight) -> point

  返回碰撞的坐标

  掩码比较：将原图转为为位图，按位相与，比较的速度较快

  需要有mask属性来保存该精灵的位图

  生成位图的方法

  sprite.mask = pygame.mask.from_surface(sprite.image)

精灵组

sprite模块的其他方法：精灵与精灵组碰撞检测

- pygame.sprite.(sprite, group, dokill, collided = None) -> Sprite_list

  返回一个组中与该精灵碰撞的 精灵

  sprite: 待检测的精灵

  group：待检测的组

  dokill: 是否杀死（移除）True 将这些精灵从组中删除

  collided：回调函数，用来选择以哪种方法进行碰撞检测

  可调用的碰撞回调函数：

  collide_rect（默认），collide_rect_ratio, collide_circle,

  collide_circle_ratio, collide_mask

精灵组

sprite模块的其他方法：精灵与精灵组碰撞检测

- pygame.sprite. spritecollide(sprite, group, dokill, collided = None) -> Sprite_list

  返回一个组中与该精灵碰撞的 精灵

  sprite: 待检测的精灵

  group：待检测的组

  dokill: 是否杀死（移除）True 将这些精灵从组中删除

  collided：回调函数，用来选择以哪种方法进行碰撞检测

  可调用的碰撞回调函数：

  collide_rect（默认），collide_rect_ratio, collide_circle,

  collide_circle_ratio, collide_mask

精灵组

sprite模块的其他方法：精灵组与精灵组碰撞检测

- pygame.sprite. groupcollide(group1, group2, dokill1, dokill2, collided = None) -> Sprite_dict

找到两个组之间发生碰撞放精灵

返回值：字典 键是组1的精灵 键2是与group2中与该精灵碰撞的精灵的列表