

坦克大战游戏

Pygame 开发基础

Pygame 游戏引擎的安装

1. doc 窗口直接使用 pip 安装

Windows+R→cmd→输入如下命令：

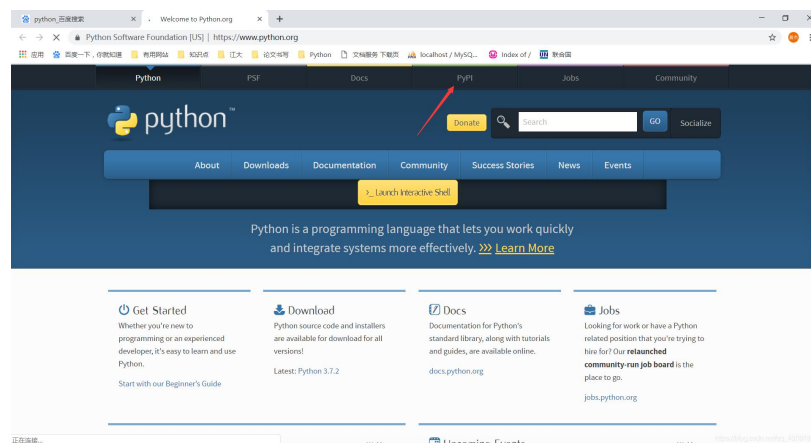
```
pip install pygame
```

或者安装指定版本

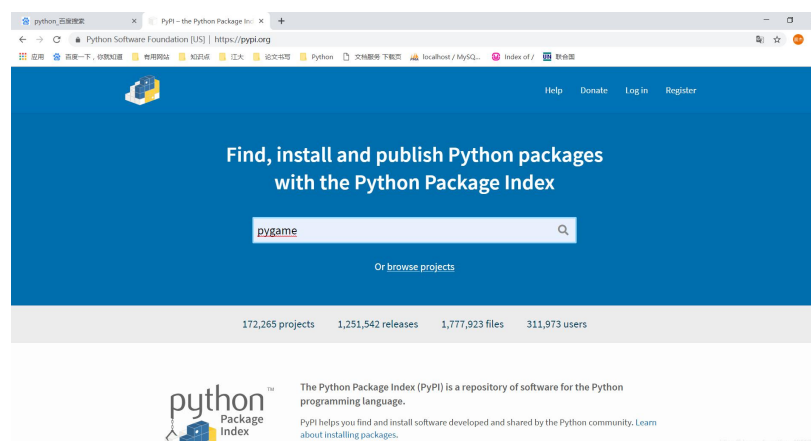
```
pip install pygame ==版本号
```

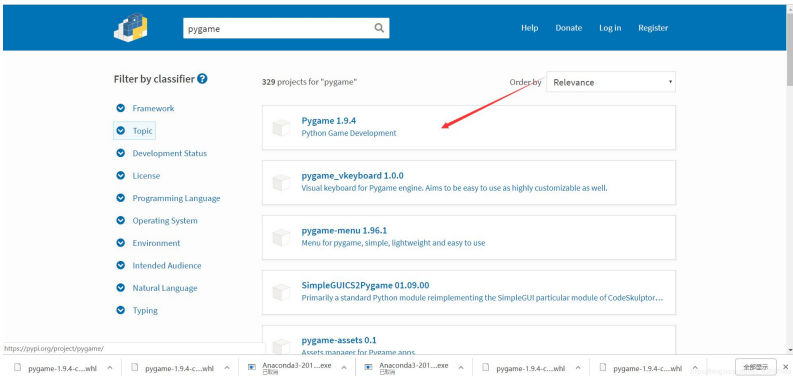
2. 下载 whl 文件进行安装

- (1) 进入 python 官网 <https://www.python.org> 点击菜单 PyPI，如下图：



- (2) 输入 pygame





(3) 点击去之后再点击左侧的 Download files。

| | | | | |
|------------------------------|--|--------|------|--------------|
| License: LGPL | pygame-1.9.6-cp35-cp35m-manylinux1_x86_64.whl (11.4 MB) SHA256 | Wheel | cp35 | Apr 25, 2019 |
| Author: A community project. | pygame-1.9.6-cp35-cp35m-win32.whl (4.0 MB) SHA256 | Wheel | cp35 | Apr 25, 2019 |
| Maintainers | pygame-1.9.6-cp35-cp35m-win_amd64.whl (4.3 MB) SHA256 | Wheel | cp35 | Apr 25, 2019 |
| illum | pygame-1.9.6-cp36-cp36m-macosx10_11_intel.whl (4.9 MB) SHA256 | Wheel | cp36 | Apr 25, 2019 |
| pygameci | pygame-1.9.6-cp36-cp36m-manylinux1_i686.whl (11.0 MB) SHA256 | Wheel | cp36 | Apr 25, 2019 |
| takowl | pygame-1.9.6-cp36-cp36m-manylinux1_x86_64.whl (11.4 MB) SHA256 | Wheel | cp36 | Apr 25, 2019 |
| | pygame-1.9.6-cp36-cp36m-win32.whl (4.0 MB) SHA256 | Wheel | cp36 | Apr 25, 2019 |
| | pygame-1.9.6-cp36-cp36m-win_amd64.whl (4.3 MB) SHA256 | Wheel | cp36 | Apr 25, 2019 |
| | pygame-1.9.6-cp37-cp37m-macosx10_11_intel.whl (4.9 MB) SHA256 | Wheel | cp37 | Apr 25, 2019 |
| | pygame-1.9.6-cp37-cp37m-manylinux1_i686.whl (11.0 MB) SHA256 | Wheel | cp37 | Apr 25, 2019 |
| | pygame-1.9.6-cp37-cp37m-manylinux1_x86_64.whl (11.4 MB) SHA256 | Wheel | cp37 | Apr 25, 2019 |
| | pygame-1.9.6-cp37-cp37m-win32.whl (4.0 MB) SHA256 | Wheel | cp37 | Apr 25, 2019 |
| | pygame-1.9.6-cp37-cp37m-win_amd64.whl (4.3 MB) SHA256 | Wheel | cp37 | Apr 25, 2019 |
| | pygame-1.9.6tar.gz (3.2 MB) SHA256 | Source | None | Apr 25, 2019 |

- (4) 找到对应的版本，进行下载。例如我下载到 E: /
- (5) 进入到 E 盘，执行此文件，输入如下命令

```
pip install pygame-1.9.6-cp36-cp36m-win_amd64.whl
```

- (6) 验证是否安装成功 在 py 文件中导入 pygame

```
import pygame
```

Pygame 框架中的模块

在 Pygame 框架中有很多模块，官方网址 <http://pygame.org/> 。其中最常用模块的具体说明如表 1-1 所示。

表 1-1 Pygame 框架中的常用模块

| 模块名 | 功能说明 |
|----------------|----------|
| pygame.display | 访问显示设备 |
| pygame.draw | 绘制形状、线和点 |
| pygame.event | 管理事件 |
| pygame.font | 使用字体 |
| pygame.image | 加载和存储图片 |
| pygame.key | 读取键盘按键 |

| | |
|-----------------|----------|
| pygame.mixer | 声音 |
| pygame.mouse | 鼠标 |
| pygame.movie | 播放视频 |
| pygame.music | 播放音频 |
| pygame.overlay | 访问高级视频叠加 |
| pygame.rect | 管理矩形区域 |
| pygame.sndarray | 操作声音数据 |
| pygame.sprite | 操作移动图像 |
| pygame.surface | 管理图像和屏幕 |

【示例 1-1】开发第一个 Pygame 程序

```
import pygame
#初始函数，使用 pygame 的第一步；
pygame.init()
#生成主屏幕 screen
screen=pygame.display.set_mode((600,500),0,32)
#设置标题
pygame.display.set_caption('Hello Pygame')
while True:
    #刷新屏幕
    pygame.display.update()
```

执行结果如图 1-1 所示：

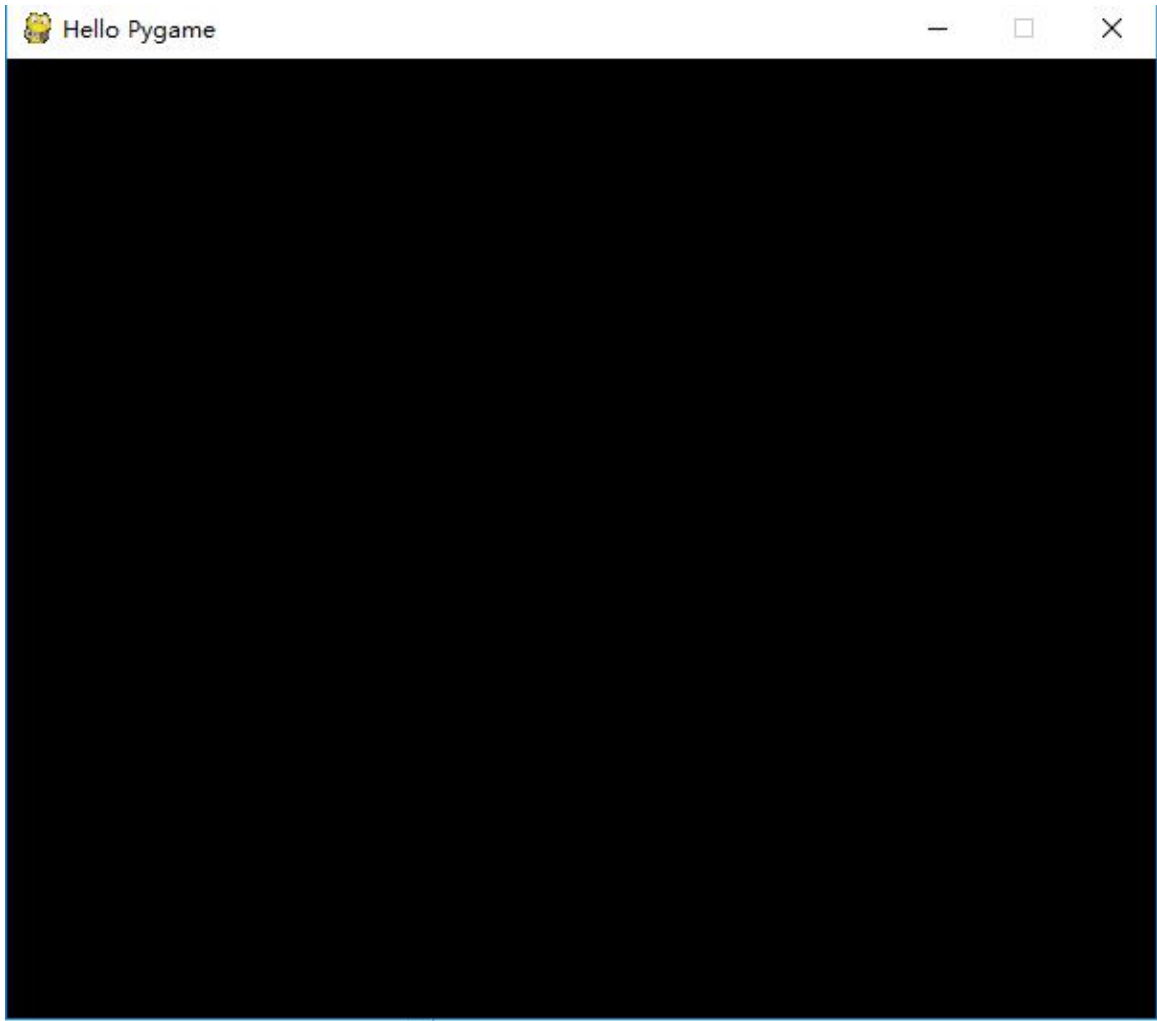


图 1-1 示例 1-1 运行效果图

对上述示例代码的具体说明如下所示。

（1）`set_mode` 函数：会返回一个 `Surface` 对象，代表了在桌面上出现的那个窗口。在 3 个参数中，第 1 个参数为元组，表示屏幕的大小；第 2 个标志位，具体含义如表 1-2 所示，如果不用什么特性，就指定 0；第 3 个为色深。

表 1-2 各个标志位的具体含义

| 标志位 | 含义 |
|------------|---|
| FULLSCREEN | 创建一个全屏窗口 |
| DOUBLEBUF | 创建一个“双缓冲”窗口，建议和 <code>HWSURFACE</code> 和 <code>OPENGL</code> 同时使用 |
| NOFRAME | 创建一个没有边框的窗口 |
| RESIZEBLE | 创建一个可以改变大小的窗口 |
| OPENGL | 创建一个 <code>OPENGL</code> 渲染的窗口 |
| HWSURFACE | 创建一个硬件加速的窗口，必须和 <code>FULLSCREEN</code> 同时使用 |

（2）游戏的主循环是一个无限循环，直到用户退出。在这个主循环里面做的事情就是

不停的刷新新画面。

注意：

如果未安装 `pygame` 模块的，打开控制台执行 `pip install pygame` 命令进行安装。

事件操作

事件是一个操作，通常来说，`Pygame` 会接受用户的各种操作（比如按键盘，移动鼠标等）。这些操作会产生对应的事件，例如按键盘事件，移动鼠标事件。事件在软件开发中非常重要，`Pygame` 把一系列的事件存放在一个队列里，并逐个进行处理。

1. 事件检索

使用函数 `pygame.event.get()` 获取所有的事件，表 1-3 列出了 `Pygame` 中常用的事件。

表 1-3 `Pygame` 中常用的事件

| 事件 | 参数 | 产生途径 |
|-----------------|-------------------|------------------------------------|
| QUIT | none | 用户按下关闭按钮 |
| ACTIVEEVENT | gain, state | 激活或者隐藏 <code>Pygame</code> |
| KEYDOWN | unicode, key, mod | 按下键 |
| KEYUP | key, mod | 放开键 |
| MOUSEMOTION | pos, rel, buttons | 鼠标移动 |
| MOUSEBUTTONUP | pos, button | 放开鼠标键 |
| MOUSEBUTTONDOWN | pos, button | 按下鼠标键 |
| JOYBUTTONUP | joy, button | 游戏手柄放开 |
| JOYBUTTONDOWN | joy, button | 游戏手柄按下 |
| VIDEORESIZE | size, w, h | <code>Pygame</code> 窗口缩放 |
| VIDEOEXPOSE | none | <code>Pygame</code> 窗口部分公开（expose） |
| USEREVENT | code | 触发一个用户事件 |

2. 处理鼠标事件

在 `Pygame` 框架中，`MOUSEMOTION` 事件会在鼠标动作的时候发生，它有如下所示 3 个参数。

- `buttons`：一个含有 3 个数字的元组，3 个值分别代表左键、中键和右键，1 就表示按下。
- `pos`：位置
- `rel`：代表现在距离上次产生鼠标事件时的距离

和 `MOUSEMOTION` 类似，常用的鼠标事件还有 `MOUSEBUTTONUP` 和 `MOUSEBUTTONDOWN` 两个。在很多时候，开发者只需要知道鼠标按下就可以不



用上面那个比较强大的事件了。这两个事件的参数如下所示。

- button: 这个值代表操作哪个按键
- pos: 位置

3. 处理键盘事件

在 Pygame 框架中, 键盘和游戏手柄的事件比较类似, 处理键盘的事件为 KEYDOWN 和 KEYUP。KEYDOWN 和 KEYUP 事件的参数描述如下所示。

- key: 按下或者放开的键值, 是一个数字, 因为很少有人可以记住, 所以在 Pygame 中可以使用 K_xxx 来表示, 比如字母 a 就是 K_a, 还有 K_SPACE 和 K_RETURN 等。
- mod: 包含了组合键信息, 如果 mod&KMOD_CTRL 是真, 表示用户同时按下了 Ctrl 键, 类似的还有 KMOD_SHIFT 和 KMOD_ALT。
- unicode: 代表了按下键对应的 Unicode 值。

【示例 1-2】处理键盘事件

```
import pygame
pygame.init()
screen=pygame.display.set_mode((600,500),0,32)
#加载图片
image=pygame.image.load('img/y.jpg')
#设置 x, y 初始值作为初始位置
x,y=0,0
#设置横向和纵向两个方向的移动距离
while True:
    for event in pygame.event.get():
        #如果是退出
        if event.type==pygame.QUIT:
            exit()
        #如果是按下
        if event.type==pygame.KEYDOWN:
            #如果按下的是左键
            if event.key==pygame.K_LEFT:
                x-=10
            elif event.key==pygame.K_RIGHT:
                x+=10
            elif event.key==pygame.K_UP:
                y-=10
```

```
elif event.key==pygame.K_DOWN:  
    y+=10  
    screen.fill((0,0,0))  
    screen.blit(image,(x,y))  
    pygame.display.update()
```

执行结果如图 1-2 所示：

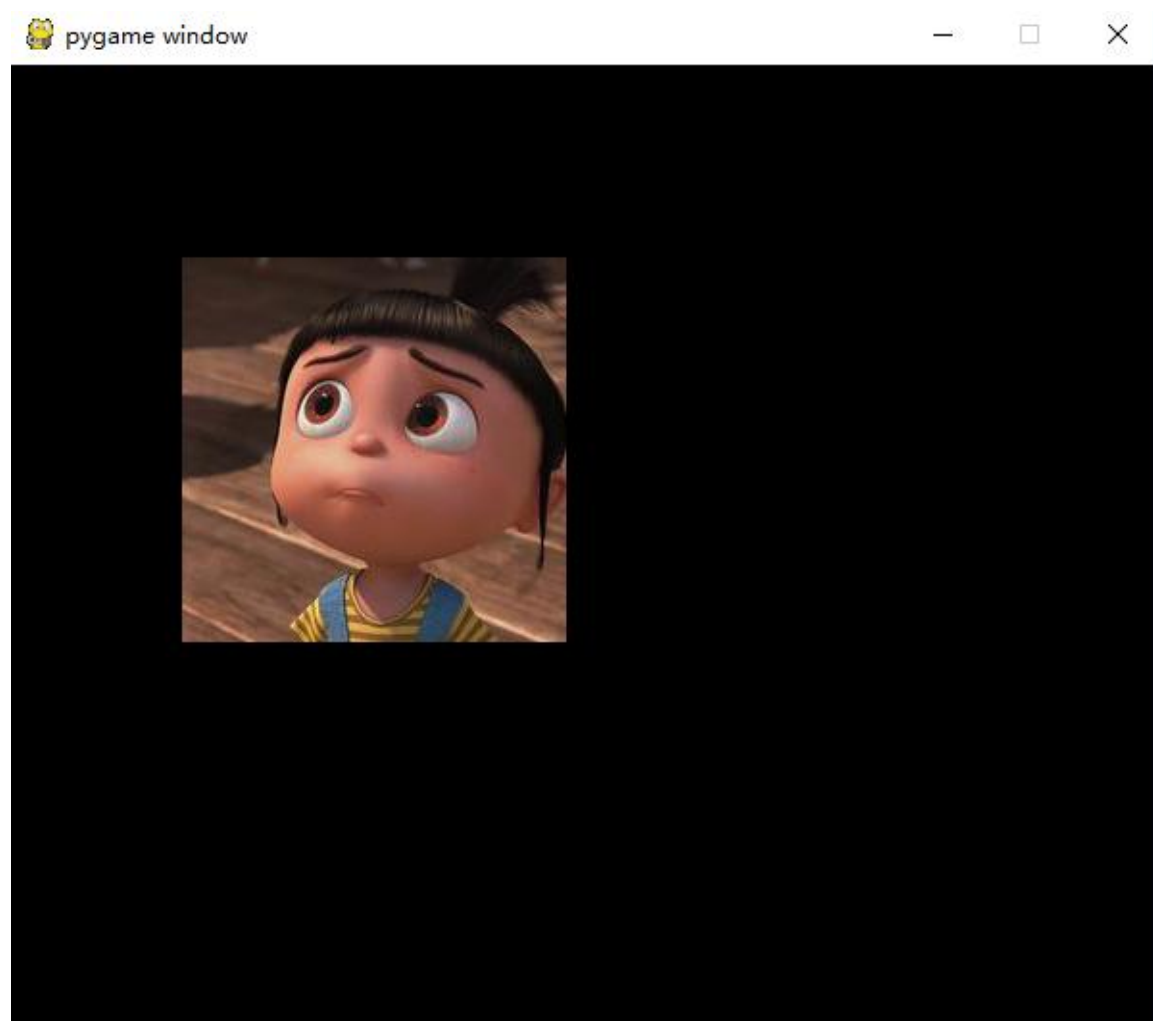


图 1-2 示例 1-2 运行效果图

字体处理

在 Pygame 模块中可以直接调用系统字体，或者可以直接使用 TTF 字体。为了使用字体，需要先创建一个 Font 对象。对系统自带的字体来说，可以使用如下代码创建一个 Font 对象。

```
font = pygame.font.SysFont('arial',18)
```

在上述代码中，第一个参数是字体名，第二个参数表示大小。一般来说，“Arial”字体在很多系统都是存在的，如果找不到，就会使用一个默认字体，这个默认字体和每个操作系

统相关。也可以使用 `pygame.font.get_fonts()` 函数来获取当前系统中所有可用的字体。

一旦创建了一个 `font` 对象，就可以通过如下代码使用 `render` 方法来写字，并且可以显示到屏幕中。

```
COLOR_RED = pygame.Color(255, 0, 0)
textSurface = font.render('Hello Pygame', True, COLOR_RED)
```

在上述代码中，第一个参数是写的文字，第二个参数是布尔值，它控制是否开启抗锯齿功能，如果设置为 `True` 字体比较平滑，不过相应的速度会有点点影响；第三个参数是字体的颜色；第四个是背景色，如果你想没有背景色，那么可以不加第四个参数。

【示例 1-3】显示指定样式的文字

```
import pygame
pygame.init()
#创建窗口
screen=pygame.display.set_mode((600,500),0,32)
#设置标题
pygame.display.set_caption('字体处理')
#创建 font 对象
font=pygame.font.SysFont('kaiti',20)
#设置文本内容和颜色
COLOR_RED=pygame.Color(255,255,255)
textSurface=font.render('你好， Pygame',True,COLOR_RED)
#设置文字的坐标
x,y=5,5
#游戏主循环
while True:
    #将文件添加到窗口
    screen.blit(textSurface,(x,y))
    pygame.display.update()
```

执行结果如图 1-3 所示：

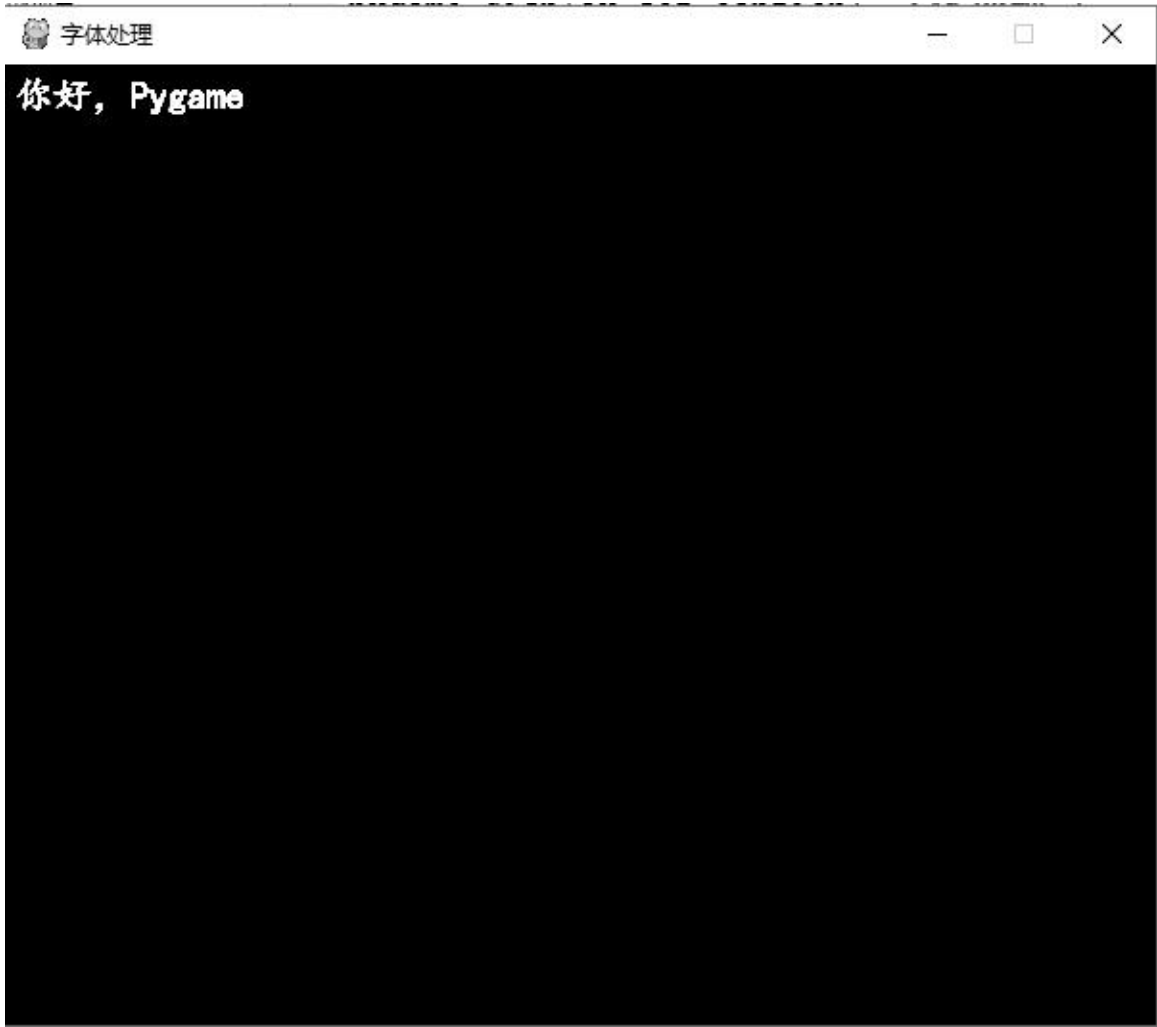


图 1-3 示例 1-3 运行效果图

绘制图形

Pygame 的坐标原点 (0, 0) 点位于左上角，X 轴自左向右，Y 轴自上向下，单位为像素。绘制图形的方法如表 1-4。

表 1-4 绘制图形

| 方法名 | 说明 |
|--|-----------|
| <code>pygame.draw.line(Surface, color, start_pos, end_pos, width)</code> | 绘制一条线段 |
| <code>pygame.draw.aaline(Surface, color, start_pos, end_pos, blend)</code> | 绘制一条抗锯齿的线 |
| <code>pygame.draw.lines(Surface, color, closed, pointlist, width)</code> | 绘制一条折线 |
| <code>pygame.draw.rect(Surface, color, Rect)</code> | 绘制一个矩形 |
| <code>pygame.draw.rect(Surface, color, Rect, width)</code> | 绘制一个矩形框 |
| <code>pygame.draw.ellipse(Surface, color, Rect)</code> | 绘制一个椭圆 |
| <code>pygame.draw.ellipse(Surface, color, Rect, width)</code> | 绘制一个多边形 |
| <code>pygame.draw.arc(Surface, color, Rect, start_angle, stop_angle, width)</code> | 绘制一条弧线 |

| | |
|--|-------|
| pygame.draw.circle(Surface, color, Rect, radius) | 绘制一个圆 |
|--|-------|

【示例 1-4】绘制图形

```
# 导入需要的模块
import pygame, sys
from pygame.locals import *
from math import pi

# 初始化 pygame
pygame.init()

# 设置窗口的大小，单位为像素
screen = pygame.display.set_mode((400, 300))

# 设置窗口标题
pygame.display.set_caption('Drawing')

# 定义颜色
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)

# 设置背景颜色
screen.fill(WHITE)

# 绘制一条线
pygame.draw.line(screen, GREEN, [0, 0], [50, 30], 5)

# 绘制一条抗锯齿的线
pygame.draw.aaline(screen, GREEN, [0, 50], [50, 80], True)

# 绘制一条折线
pygame.draw.lines(screen, BLACK, False,
                  [[0, 80], [50, 90], [200, 80], [220, 30]], 5)
```

```
# 绘制一个空心矩形
pygame.draw.rect(screen, BLACK, [75, 10, 50, 20], 2)

# 绘制一个矩形
pygame.draw.rect(screen, BLACK, [150, 10, 50, 20])

# 绘制一个空心椭圆
pygame.draw.ellipse(screen, RED, [225, 10, 50, 20], 2)

# 绘制一个椭圆
pygame.draw.ellipse(screen, RED, [300, 10, 50, 20])

# 绘制多边形
pygame.draw.polygon(screen, BLACK, [[100, 100], [0, 200], [200, 200]], 5)

# 绘制多条弧线
pygame.draw.arc(screen, BLACK, [210, 75, 150, 125], 0, pi / 2, 2)
pygame.draw.arc(screen, GREEN, [210, 75, 150, 125], pi / 2, pi, 2)
pygame.draw.arc(screen, BLUE, [210, 75, 150, 125], pi, 3 * pi / 2, 2)
pygame.draw.arc(screen, RED, [210, 75, 150, 125], 3 * pi / 2, 2 * pi, 2)

# 绘制一个圆
pygame.draw.circle(screen, BLUE, [60, 250], 40)

# 程序主循环
while True:
    # 获取事件
    for event in pygame.event.get():
        # 判断事件是否为退出事件
        if event.type == QUIT:
            # 退出 pygame
            pygame.quit()
            # 退出系统
            sys.exit()
```

```
# 绘制屏幕内容
pygame.display.update()
```

执行结果如图 1-3 所示：

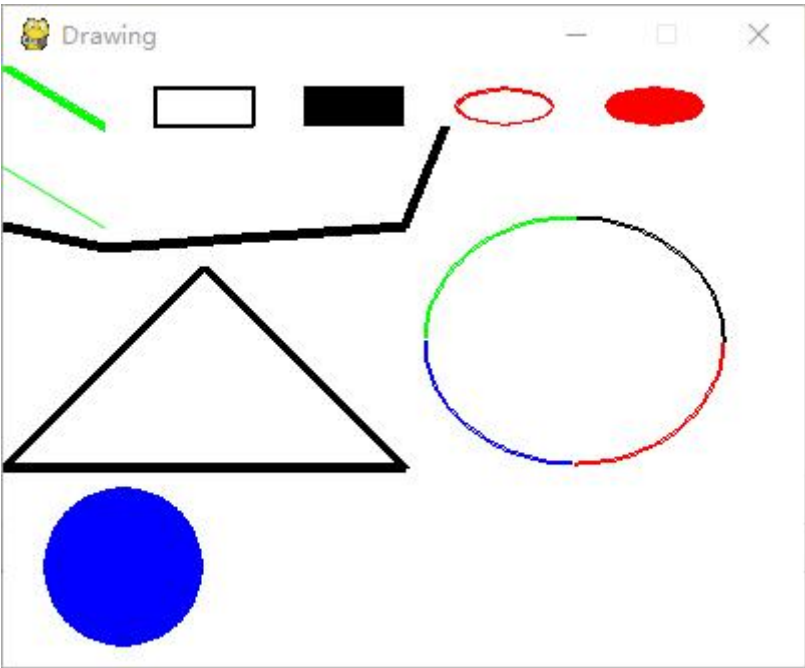


图 1-4 示例 1-4 运行效果图

实现动画

由于人类眼睛的特殊生理结构，当所看画面的帧率高于 24 的时候，就会认为是连贯的，此现象称之为 视觉暂留 。

帧率 (Frame rate) 是用于测量显示帧数的量度，所谓的测量单位为每秒显示帧数 (Frames per Second，简称：FPS) 。

一般来说 30fps 是可以接受的，但是将性能提升至 60fps 则可以明显提升交互感和逼真感，但是一般来说超过 75fps 一般就不容易察觉到有明显的流畅度提升了。

在我们原有坐标系的基础上添加偏移量，再重新绘制，依次一张一张的循环绘制下去，就会得到我们想要的物体移动的效果。

Pygame 实现动画主要用到的方法如表 1-5 所示：

表 1-5 实现动画主要方法

| 方法名 | 说明 |
|---|------------------------------|
| pygame.image.load(filename) | 加载一张图片 |
| pygame.Surface.blit(source, dest, area=None, special_flags = 0) | 将图片绘制到屏幕相应坐标上（后面两个参数默认，可以不传） |
| pygame.time.Clock() | 获得 pygame 的时钟 |
| pygame.time.Clock.tick(FPS) | 设置 pygame 时钟的间隔时间 |

【示例 1-5】实现动画

```
# 导入需要的模块
import pygame, sys
from pygame.locals import *

# 初始化 pygame
pygame.init()

# 设置帧率（屏幕每秒刷新的次数）
FPS = 30

# 获得 pygame 的时钟
fpsClock = pygame.time.Clock()

# 设置窗口大小
screen = pygame.display.set_mode((500, 400), 0, 32)

# 设置标题
pygame.display.set_caption('Animation')

# 定义颜色
WHITE = (255, 255, 255)

# 加载一张图片（所用到的的图片请参考 1.5 代码获取）
img = pygame.image.load('img/donghua.jpg')

# 初始化图片的位置
imgx = 10
imgy = 10

# 初始化图片的移动方向
direction = 'right'

# 程序主循环
while True:
```

```
# 每次都要重新绘制背景白色
screen.fill(WHITE)

# 判断移动的方向，并对相应的坐标做加减
if direction == 'right':
    imgx += 5
    if imgx == 380:
        direction = 'down'
elif direction == 'down':
    imgy += 5
    if imgy == 300:
        direction = 'left'
elif direction == 'left':
    imgx -= 5
    if imgx == 10:
        direction = 'up'
elif direction == 'up':
    imgy -= 5
    if imgy == 10:
        direction = 'right'

# 该方法将用于图片绘制到相应的坐标中
screen.blit(img, (imgx, imgy))

for event in pygame.event.get():
    if event.type == QUIT:
        pygame.quit()
        sys.exit()

# 刷新屏幕
pygame.display.update()

# 设置 pygame 时钟的间隔时间
fpsClock.tick(FPS)
```

执行结果如图 1-5 所示：

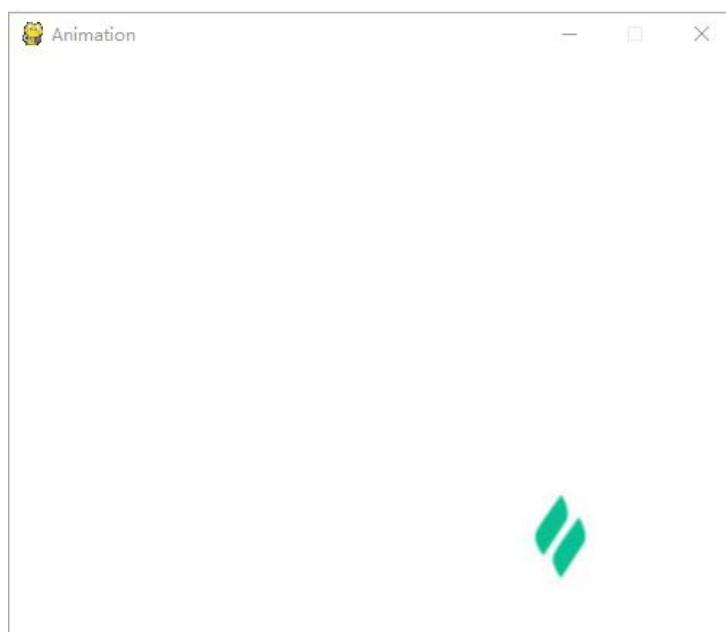


图 1-5 示例 1-5 运行效果图

播放音乐

在 Pygame 里播放音频有两个方法，一个用来播放特效声音，一个用来播放背景音乐：

```
pygame.mixer.Sound(filename)
```

该方法返回一个 Sound 对象，调用它的.play()方法，即可播放较短的音频文件（比如玩家受到伤害、收集到金币等）；

```
pygame.mixer.music.load(filename)
```

该方法用来加载背景音乐，之后调用 pygame.mixer.music.play()方法就可以播放背景音乐（Pygame 只允许加载一个背景音乐在同一个时刻）

【示例 1-5】实现动画

```
# 导入需要的模块
import pygame, sys
from pygame.locals import *

# 初始化 pygame
pygame.init()

# 设置窗口的大小，单位为像素
```

```
screen = pygame.display.set_mode((500, 400))

# 设置窗口的标题
pygame.display.set_caption('Audio')

# 定义颜色
WHITE = (255, 255, 255)

# 设置背景
screen.fill(WHITE)

# 加载并播放一个特效音频文件
sound = pygame.mixer.Sound('img/start.wav')
sound.play()

# 加载背景音乐文件
pygame.mixer.music.load('img/fire.wav')

# 播放背景音乐，第一个参数为播放的次数（-1 表示无限循环），第二个参数是设置播放
的起点（单位为秒）
pygame.mixer.music.play(-1, 0.0)

# 程序主循环
while True:

    # 获取事件
    for event in pygame.event.get():
        # 判断事件是否为退出事件
        if event.type == QUIT:
            # 停止播放背景音乐
            pygame.mixer.music.stop()

            # 退出 pygame
            pygame.quit()

            # 退出系统
            sys.exit()
```



```
# 绘制屏幕内容
pygame.display.update()
```

坦克大战游戏开发

《坦克大战》是由 Namco 游戏公司开发的一款平面射击游戏，于 1985 年发售。游戏以坦克战斗及保卫基地为主题，属于策略型联机类。同时也是 FC 平台上少有的内建关卡编辑器的几个游戏之一，玩家可自己创建独特的关卡，并通过获取一些道具使坦克和基地得到强化。它看似简单但变化无穷，令人上瘾。本节将介绍使用“Python+Pygame”开发一个简单坦克大战游戏的方法，并详细介绍其具体的实现流程。

项目搭建

本游戏主要分为两个对象，分别是我方坦克和敌方坦克。用户可以通过控制我方的坦克来摧毁敌方的坦克保护自己的“家”，把所有的敌方坦克消灭完达到胜利。敌方的坦克在初始的时候是默认 5 个的（这可以自己设置），当然，如果我方坦克被敌方坦克的子弹打中，游戏结束。从面向对象分析该项目有以下类组成：

- 主类：主要包括开始游戏、结束游戏的功能。

```
class MainGame():
    #开始游戏方法
    def startGame(self):
        pass
    def endGame(self):
        pass
```

- 坦克类：主要包括坦克的创建、显示、移动及射击的功能。

```
class Tank():
    def __init__(self):
        pass
    #坦克的移动方法
    def move(self):
        pass
    #碰撞墙壁的方法
    def hitWalls(self):
        pass
    #射击方法
```

```
def shot(self):
    pass
#展示坦克
def displayTank(self):
    pass
```

- 我方坦克类继承坦克类，主要包括创建、与敌方坦克的碰撞方法。

```
class MyTank(Tank):
    def __init__(self):
        pass
#碰撞敌方坦克的方法
def hitEnemyTank(self):
    pass
```

- 敌方坦克类继承坦克类，主要包括创建、与我方坦克碰撞方法。

```
class EnemyTank(Tank):
    def __init__(self):
        pass
def hitMyTank(self):
    pass
```

- 子弹类：主要包括子弹的创建、显示及移动的功能。

```
class Bullet():
    def __init__(self):
        pass
#子弹的移动方法
def bulletMove(self):
    pass
#展示子弹的方法
def displayBullet(self):
    pass
#我方子弹碰撞敌方坦克的方法
def hitEnemyTank(self):
    pass
#敌方子弹与我方坦克的碰撞方法
```

```
def hitMyTank(self):
    pass
#子弹与墙壁的碰撞
def hitWalls(self):
    pass
```

- 墙壁类：主要包括墙壁的创建、显示的功能。

```
class Wall():
    def __init__(self):
        pass
#展示墙壁的方法
def displayWall(self):
    pass
```

- 爆炸效果类：主要展示爆炸效果。

```
class Explode():
    def __init__(self):
        pass
#展示爆炸效果
def displayExplode(self):
    pass
```

- 音效类：主要播放音乐。

```
class Music():
    def __init__(self):
        pass
#开始播放音乐
def play(self):
    pass
```

显示游戏窗口

在游戏设计的前期，要先创建游戏的界面，也就是要为所设计的游戏创建一个窗口，可以通过如下代码实现：

【示例 1-4】显示游戏窗口

```
import pygame
```

```
_display = pygame.display
COLOR_BLACK = pygame.Color(0, 0, 0)
class MainGame():
    #游戏主窗口
    window = None
    SCREEN_HEIGHT = 500
    SCREEN_WIDTH = 800
    def __init__(self):
        pass
    #开始游戏方法
    def startGame(self):
        _display.init()
        #创建窗口加载窗口
        MainGame.window =
_display.set_mode([MainGame.SCREEN_WIDTH,MainGame.SCREEN_HEIGHT])
        #设置一下游戏标题
        _display.set_caption("坦克大战 v1.03")
        #让窗口持续刷新操作
        while True:
            #给窗口完成一个填充颜色
            MainGame.window.fill(COLOR_BLACK)
            #窗口的刷新
            _display.update()
MainGame().startGame()
```

执行结果如图 1-4 所示：



图 1-4 示例 1-4 运行效果图

添加提示文字

在运行代码时会发现，创建的窗口没有任何提示。然而在实际中希望窗口提示敌方坦克的数量，因此，需要在现有窗口进行必须的改进，添加敌方坦克数量提示。

添加左上角提示文字。

```
#左上角文字绘制功能
def getTextSurface(self,text):
    # 初始化字体模块
    pygame.font.init()
    # 选中一个合适的字体
    font = pygame.font.SysFont('kaiti',18)
    # 使用对应的字符完成相关内容的绘制
    textSurface = font.render(text,True,COLOR_RED)
    return textSurface
```

在开始游戏方法中调用添加提示文字方法。

```
import pygame
_display = pygame.display
COLOR_BLACK = pygame.Color(0, 0, 0)
```

```
COLOR_RED = pygame.Color(255, 255, 255)
version = 'v1.05'
class MainGame():
    #游戏主窗口
    window = None
    SCREEN_HEIGHT = 500
    SCREEN_WIDTH = 800
    def __init__(self):
        pass
    #开始游戏方法
    def startGame(self):
        _display.init()
        #创建窗口加载窗口
        MainGame.window
    _display.set_mode([MainGame.SCREEN_WIDTH,MainGame.SCREEN_HEIGHT])
    #设置一下游戏标题
    _display.set_caption("坦克大战 v1.03")
    #让窗口持续刷新操作
    while True:
        #给窗口完成一个填充颜色
        MainGame.window.fill(COLOR_BLACK)
        #将绘制文字得到的小画布，粘贴到窗口中
        MainGame.window.blit(self.getTextSurface("剩余敌方坦克%d 辆"%5),(5,5))
        #窗口的刷新
        _display.update()
#调用开始游戏的方法
MainGame().startGame()
```

执行结果如图 1-5 所示：



图 1-5 添加提示文字运行效果图

加载我方坦克

通过加载一张图片来表示游戏中的坦克，此坦克代表我方坦克，完善坦克类。

```
class Tank():
    def __init__(self, left, top):
        self.images = {
            'U': pygame.image.load('img/p1tankU.gif'),
            'D': pygame.image.load('img/p1tankD.gif'),
            'L': pygame.image.load('img/p1tankL.gif'),
            'R': pygame.image.load('img/p1tankR.gif')
        }
        self.direction = 'U'
        self.image = self.images[self.direction]
        # 坦克所在的区域 Rect->
        self.rect = self.image.get_rect()
        # 指定坦克初始化位置 分别距 x, y 轴的位置
        self.rect.left = left
        self.rect.top = top
        # 展示坦克(将坦克这个 surface 绘制到窗口中 blit())
    def displayTank(self):
```

```
#1.重新设置坦克的图片
self.image = self.images[self.direction]

#2.将坦克加入到窗口中

MainGame.window.blit(self.image,self.rect)
```

开始游戏方法，创建坦克，将坦克添加到窗口。

```
import pygame
_display = pygame.display
COLOR_BLACK = pygame.Color(0, 0, 0)
COLOR_RED = pygame.Color(255, 255, 255)
class MainGame():
    #游戏主窗口
    window = None
    SCREEN_HEIGHT = 500
    SCREEN_WIDTH = 800
    #创建我方坦克
    TANK_P1 = None
    def __init__(self):
        pass
    #开始游戏方法
    def startGame(self):
        _display.init()
        #创建窗口加载窗口(借鉴官方文档)
        MainGame.window = pygame.display.set_mode([MainGame.SCREEN_WIDTH,MainGame.SCREEN_HEIGHT])
        #创建我方坦克
        MainGame.TANK_P1 = Tank(400,300)
        #设置一下游戏标题
        _display.set_caption("坦克大战 v1.03")
        #让窗口持续刷新操作
        while True:
            #给窗口完成一个填充颜色
            MainGame.window.fill(COLOR_BLACK)
            #将绘制文字得到的小画布，粘贴到窗口中
            MainGame.window.blit(self.getTextSurface("剩余敌方坦克%d 辆"%5),(5,5))
            #将我方坦克加入到窗口中
```



```
MainGame.TANK_P1.displayTank()  
#窗口的刷新  
_display.update()  
#调用开始游戏的方法  
MainGame().startGame()
```

执行结果如图 1-6 所示：

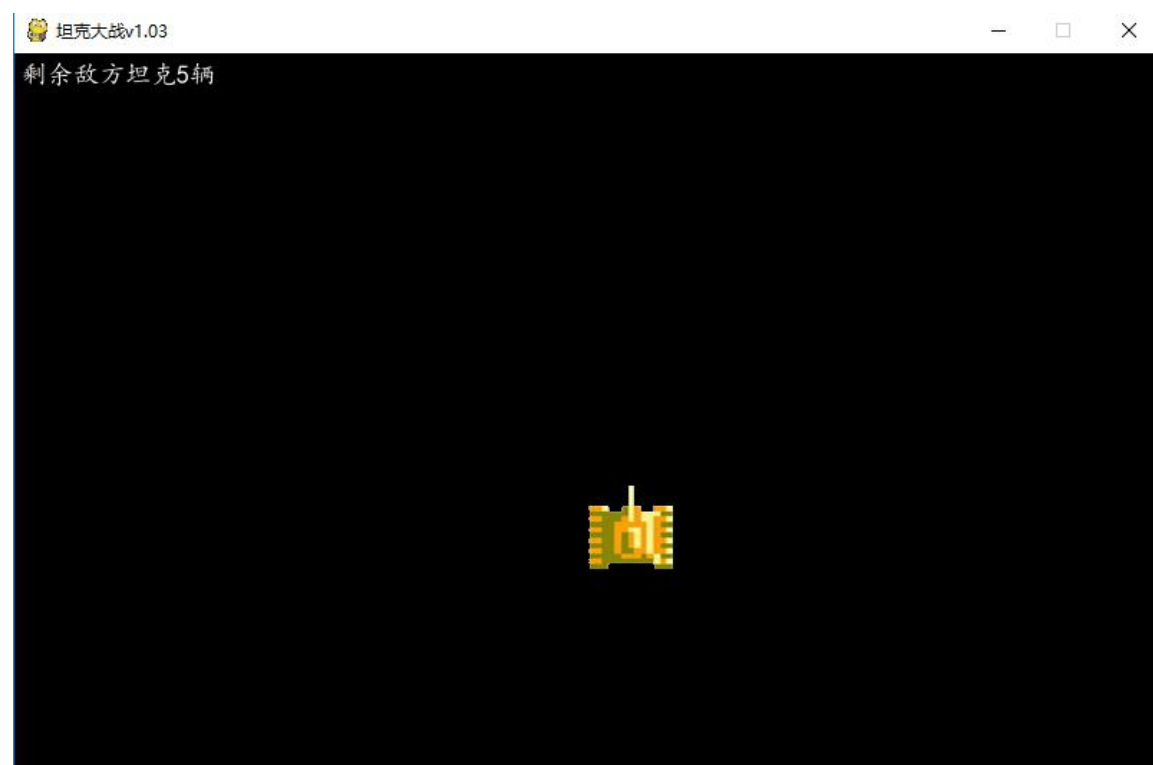


图 1-6 加载我方坦克运行效果图

添加事件监听

上面创建的坦克还不能动，显示不是创建游戏的目的，因此，要给创建的坦克赋予“生命”。添加事件监听，控制上、下、左、右四个方向键，实现针对不同的键改变坦克的方向及移动功能，点击关闭退出游戏。

实现退出方法。

```
def endGame(self):  
    print("谢谢使用")  
    #结束 python 解释器  
    exit()
```

坦克类中添加速度属性，实现坦克移动。

```

#坦克的移动方法
def move(self):
    if self.direction == 'L':
        if self.rect.left > 0:
            self.rect.left -= self.speed
    elif self.direction == 'R':
        if self.rect.left + self.rect.height < MainGame.SCREEN_WIDTH:
            self.rect.left += self.speed
    elif self.direction == 'U':
        if self.rect.top > 0:
            self.rect.top -= self.speed
    elif self.direction == 'D':
        if self.rect.top + self.rect.height < MainGame.SCREEN_HEIGHT:
            self.rect.top += self.speed

```

坦克类中添加移动开关属性，按下上、下、左、右四个方向键修改坦克的方向及开关状态，按下关闭键，调用关闭方法退出游戏。

#获取程序期间所有事件(鼠标事件， 键盘事件)

```

def getEvent(self):
    #1.获取所有事件
    eventList = pygame.event.get()
    #2.对事件进行判断处理(1、点击关闭按钮 2、按下键盘上的某个按键)
    for event in eventList:
        #判断 event.type 是否 QUIT，如果是退出的话，直接调用程序结束方法
        if event.type == pygame.QUIT:
            self.endGame()
        #判断事件类型是否为按键按下，如果是，继续判断按键是哪一个按键，来进行对应的处理
        if event.type == pygame.KEYDOWN:
            #具体是哪一个按键的处理
            if event.key == pygame.K_LEFT:
                print("坦克向左调头，移动")
                #修改坦克方向
                MainGame.TANK_P1.direction = 'L'
                MainGame.TANK_P1.stop = False

```

```

elif event.key == pygame.K_RIGHT:
    print("坦克向右调头, 移动")
    # 修改坦克方向
    MainGame.TANK_P1.direction = 'R'
    MainGame.TANK_P1.stop = False
elif event.key == pygame.K_UP:
    print("坦克向上调头, 移动")
    # 修改坦克方向
    MainGame.TANK_P1.direction = 'U'
    MainGame.TANK_P1.stop = False
elif event.key == pygame.K_DOWN:
    print("坦克向下掉头, 移动")
    # 修改坦克方向
    MainGame.TANK_P1.direction = 'D'
    MainGame.TANK_P1.stop = False
elif event.key == pygame.K_SPACE:
    print("发射子弹")
#结束游戏方法
if event.type == pygame.KEYUP:
    #松开的如果是方向键, 才更改移动开关状态
    if event.key == pygame.K_LEFT or event.key == pygame.K_RIGHT or
event.key == pygame.K_UP or event.key == pygame.K_DOWN:
        # 修改坦克的移动状态
        MainGame.TANK_P1.stop = True

```

随机生成敌方坦克

初始化敌方坦克。

```

class EnemyTank(Tank):
    def __init__(self, left, top, speed):
        self.images = {
            'U': pygame.image.load('img/enemy1U.gif'),
            'D': pygame.image.load('img/enemy1D.gif'),
            'L': pygame.image.load('img/enemy1L.gif'),
            'R': pygame.image.load('img/enemy1R.gif')
        }

```

```

self.direction = self.randDirection()
self.image = self.images[self.direction]
# 坦克所在的区域 Rect->
self.rect = self.image.get_rect()
# 指定坦克初始化位置 分别距 x, y 轴的位置
self.rect.left = left
self.rect.top = top
# 新增速度属性
self.speed = speed
self.stop = True

```

生成随机的四个方向。

```

def randDirection(self):
    num = random.randint(1,4)
    if num == 1:
        return 'U'
    elif num == 2:
        return 'D'
    elif num == 3:
        return 'L'
    elif num == 4:
        return 'R'

```

创建敌方坦克。

```

#创建敌方坦克
def creatEnemyTank(self):
    top = 100
    speed = random.randint(3,6)
    for i in range(MainGame.EnemTank_count):
        #每次都随机生成一个 left 值
        left = random.randint(1, 7)
        eTank = EnemyTank(left*100,top,speed)
        MainGame.EnemyTank_list.append(eTank)

```

实现敌方坦克的随机移动。

```

#随机移动

```

```
def randMove(self):
    if self.step <= 0:
        self.direction = self.randDirection()
        self.step = 50
    else:
        self.move()
        self.step -= 1
```

将敌方坦克加到窗口中。

```
#将敌方坦克加入到窗口中
def blitEnemyTank(self):
    for eTank in MainGame.EnemyTank_list:
        eTank.displayTank()
        #坦克移动的方法
        eTank.randMove()
```

在开始游戏方法，加载敌方坦克。

```
import pygame,time,random
_display = pygame.display
COLOR_BLACK = pygame.Color(0, 0, 0)
COLOR_RED = pygame.Color(255, 0, 0)
class MainGame():
    #游戏主窗口
    window = None
    SCREEN_HEIGHT = 500
    SCREEN_WIDTH = 800
    #创建我方坦克
    TANK_P1 = None
    #存储所有敌方坦克
    EnemyTank_list = []
    #要创建的敌方坦克的数量
    EnemTank_count = 5
    #存储我方子弹的列表
    Bullet_list = []
    def __init__(self):
```

```

pass

#开始游戏方法
def startGame(self):
    _display.init()
    #创建窗口加载窗口(借鉴官方文档)
    MainGame.window
    _display.set_mode([MainGame.SCREEN_WIDTH,MainGame.SCREEN_HEIGHT])
    #创建我方坦克
    MainGame.TANK_P1 = Tank(400,300)
    self.creatEnemyTank()
    #设置一下游戏标题
    _display.set_caption("坦克大战 v1.03")
    #让窗口持续刷新操作
    while True:
        #给窗口完成一个填充颜色
        MainGame.window.fill(COLOR_BLACK)
        #在循环中持续完成事件的获取
        self.getEvent()
        #将绘制文字得到的小画布，粘贴到窗口中
        MainGame.window.blit(self.getTextSurface("剩余敌方坦克%d 辆"%5),(5,5))
        #将我方坦克加入到窗口中
        MainGame.TANK_P1.displayTank()
        #循环展示敌方坦克
        self.blitEnemyTank()
        #根据坦克的开关状态调用坦克的移动方法
        if MainGame.TANK_P1 and not MainGame.TANK_P1.stop:
            MainGame.TANK_P1.move()
        time.sleep(0.02)
        #窗口的刷新
        _display.update()

```

执行结果如图 1-7 所示：

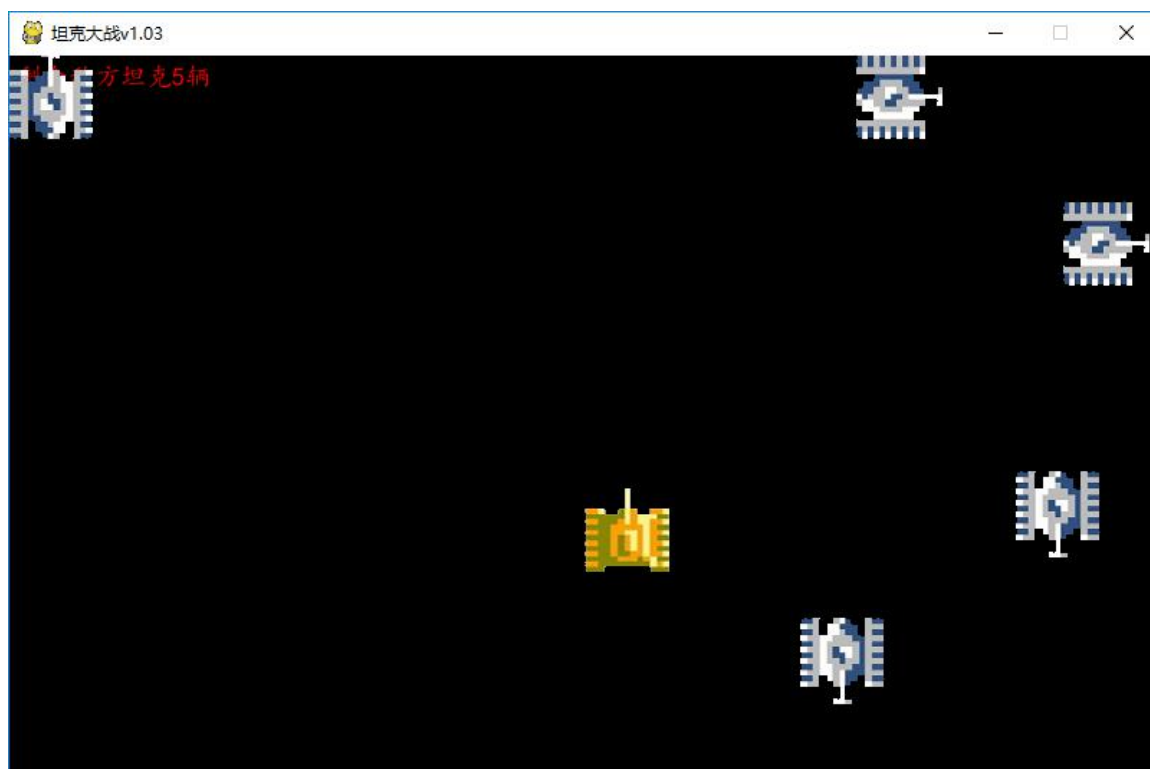


图 1-7 随机加载敌方坦克运行效果图

我方坦克发射子弹

初始化子弹。

```
def __init__(self, tank):
    #图片
    self.image = pygame.image.load('img/enemymissile.gif')
    #方向（坦克方向）
    self.direction = tank.direction
    #位置
    self.rect = self.image.get_rect()
    if self.direction == 'U':
        self.rect.left = tank.rect.left + tank.rect.width/2 - self.rect.width/2
        self.rect.top = tank.rect.top - self.rect.height
    elif self.direction == 'D':
        self.rect.left = tank.rect.left + tank.rect.width / 2 - self.rect.width / 2
        self.rect.top = tank.rect.top + tank.rect.height
    elif self.direction == 'L':
        self.rect.left = tank.rect.left - self.rect.width / 2 - self.rect.width / 2
        self.rect.top = tank.rect.top + tank.rect.width / 2 - self.rect.width / 2
```

```
elif self.direction == 'R':  
    self.rect.left = tank.rect.left + tank.rect.width  
    self.rect.top = tank.rect.top + tank.rect.width / 2 - self.rect.width / 2  
#速度  
self.speed = 7  
#用来记录子弹是否活着  
self.live = True
```

实现子弹移动。

#子弹的移动方法

```
def bulletMove(self):  
    if self.direction == 'U':  
        if self.rect.top > 0:  
            self.rect.top -= self.speed  
        else:  
            #修改状态值  
            self.live = False  
    elif self.direction == 'D':  
        if self.rect.top < MainGame.SCREEN_HEIGHT - self.rect.height:  
            self.rect.top += self.speed  
        else:  
            # 修改状态值  
            self.live = False  
    elif self.direction == 'L':  
        if self.rect.left > 0:  
            self.rect.left -= self.speed  
        else:  
            # 修改状态值  
            self.live = False  
    elif self.direction == 'R':  
        if self.rect.left < MainGame.SCREEN_WIDTH - self.rect.width:  
            self.rect.left += self.speed  
        else:  
            # 修改状态值  
            self.live = False
```


展示子弹。

```
#展示子弹的方法
def displayBullet(self):
    MainGame.window.blit(self.image,self.rect)
```

按空格键产生子弹，并将子弹添加到子弹列表中。

```
elif event.key == pygame.K_SPACE:
    print("发射子弹")
    #产生一颗子弹
    m = Bullet(MainGame.TANK_P1)
    #将子弹加入到子弹列表
    MainGame.Bullet_list.append(m)
```

将子弹添加到窗口。

```
#将我方子弹加入到窗口中
def blitBullet(self):
    for bullet in MainGame.Bullet_list:
        #如果子弹还活着，绘制出来，否则，直接从列表中移除该子弹
        if bullet.live:
            bullet.displayBullet()
            # 让子弹移动
            bullet.bulletMove()
        else:
            MainGame.Bullet_list.remove(bullet)
```

敌方坦克随机发射子弹

实现发射子弹方法。

```
def shot(self):
    num = random.randint(1,1000)
    if num <= 20:
        return Bullet(self)
```

敌方坦克加入窗口后，发射子弹，并将子弹添加到敌方子弹列表中。

```
#将敌方坦克加入到窗口中
def blitEnemyTank(self):
```

```

for eTank in MainGame.EnemyTank_list:
    eTank.displayTank()
    #坦克移动的方法
    eTank.randMove()
    #调用敌方坦克的射击
    eBullet = eTank.shot()
    #如果子弹为 None。不加入到列表
    if eBullet:
        # 将子弹存储敌方子弹列表
        MainGame.Enemy_bullet_list.append(eBullet)

```

将敌方发射的子弹添加到窗口。

```

#将敌方子弹加入到窗口中
def blitEnemyBullet(self):
    for eBullet in MainGame.Enemy_bullet_list:
        # 如果子弹还活着，绘制出来，否则，直接从列表中移除该子弹
        if eBullet.live:
            eBullet.displayBullet()
            # 让子弹移动
            eBullet.bulletMove()
        else:
            MainGame.Enemy_bullet_list.remove(eBullet)

```

我方法子弹与敌方坦克的碰撞检测

在游戏开发中，通常把显示图像的对象叫做精灵 `Spire`，精灵需要有两个属性 `image` 要显示的图像，`rect` 图像要显示在屏幕的位置。

在 Pygame 框架中，使用 `pygame.sprite` 模块中的内置函数可以实现碰撞检测。代码如下：

```
pygame.sprite.collide_rect(first, second) #返回布尔值
```

`pygame.sprite.Sprite` 是 `pygame` 精灵的基类，一般来说，总是需要写一个自己的精灵类继承 `pygame.sprite.Sprite`。让坦克类、子弹类都继承编写的精灵类。

```

class BaseItem(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)

```

```
class Tank(BaseItem):
class Bullet(BaseItem):
```

在子弹类中增加我方子弹碰撞敌方坦克的方法，如果发生碰撞，修改我方子弹及敌方坦克 live 属性的状态值。

```
#新增我方子弹碰撞敌方坦克的方法
def hitEnemyTank(self):
    for eTank in MainGame.EnemyTank_list:
        if pygame.sprite.collide_rect(eTank,self):
            self.live = False
            eTank.live = False
```

在我方子弹移动后判断子弹是否与敌方坦克碰撞。

```
#将我方子弹加入到窗口中
def blitBullet(self):
    for bullet in MainGame.Bullet_list:
        #如果子弹还活着，绘制出来，否则，直接从列表中移除该子弹
        if bullet.live:
            bullet.displayBullet()
            # 让子弹移动
            bullet.bulletMove()
            # 调用我方子弹与敌方坦克的碰撞方法
            bullet.hitEnemyTank()
        else:
            MainGame.Bullet_list.remove(bullet)
```

添加爆炸效果

初始化。

```
class Explode():
    def __init__(self,tank):
        self.rect = tank.rect
        self.step = 0
        self.images = [
            pygame.image.load('img/blast0.gif'),
            pygame.image.load('img/blast1.gif'),
            pygame.image.load('img/blast2.gif'),
            pygame.image.load('img/blast3.gif'),
```

```

        pygame.image.load('img/blast4.gif')
    ]
    self.image = self.images[self.step]
    self.live = True

```

展示爆炸效果。

```

#展示爆炸效果
def displayExplode(self):
    if self.step < len(self.images):
        MainGame.window.blit(self.image, self.rect)
        self.image = self.images[self.step]
        self.step += 1
    else:
        self.live = False
        self.step = 0

```

在我方子弹碰撞敌方坦克的方法中，如果检测到碰撞，产生爆炸类，并将爆炸效果添加到爆炸列表。

```

#新增我方子弹碰撞敌方坦克的方法
def hitEnemyTank(self):
    for eTank in MainGame.EnemyTank_list:
        if pygame.sprite.collide_rect(eTank, self):
            #产生一个爆炸效果
            explode = Explode(eTank)
            #将爆炸效果加入到爆炸效果列表
            MainGame.Explode_list.append(explode)
            self.live = False
            eTank.live = False

```

将爆炸效果添加到窗口。

```

#新增方法： 展示爆炸效果列表
def displayExplodes(self):
    for explode in MainGame.Explode_list:
        if explode.live:
            explode.displayExplode()

```

```
else:
    MainGame.Explode_list.remove(explode)
```

我方坦克的消亡

子弹类中，新增敌方子弹与我方坦克的碰撞。如果发生碰撞，修改敌方子弹、我方坦克的状态及产生爆炸效果。

```
#新增敌方子弹与我方坦克的碰撞方法
def hitMyTank(self):
    if pygame.sprite.collide_rect(self,MainGame.TANK_P1):
        # 产生爆炸效果，并加入到爆炸效果列表中
        explode = Explode(MainGame.TANK_P1)
        MainGame.Explode_list.append(explode)
        #修改子弹状态
        self.live = False
        #修改我方坦克状态
        MainGame.TANK_P1.live = False
```

添加敌方子弹到窗口中时候，如果子弹还活着，显示子弹、调用子弹移动并判断敌方子弹是否与我方坦克发生碰撞。

```
#将敌方子弹加入到窗口中
def blitEnemyBullet(self):
    for eBullet in MainGame.Enemy_bullet_list:
        # 如果子弹还活着，绘制出来，否则，直接从列表中移除该子弹
        if eBullet.live:
            eBullet.displayBullet()
            # 让子弹移动
            eBullet.bulletMove()
            if MainGame.TANK_P1 and MainGame.TANK_P1.live:
                eBullet.hitMyTank()
        else:
            MainGame.Enemy_bullet_list.remove(eBullet)
```

加载墙壁

初始化。

```
class Wall():
```

```

def __init__(self,left,top):
    self.image = pygame.image.load('img/steels.gif')
    self.rect = self.image.get_rect()
    self.rect.left = left
    self.rect.top = top
    #用来判断墙壁是否应该在窗口中展示
    self.live = True
    #用来记录墙壁的生命值
    self.hp = 3
    #展示墙壁的方法
    def displayWall(self):
        MainGame.window.blit(self.image,self.rect)

```

创建墙壁。

```

#创建墙壁的方法
def creatWalls(self):
    for i in range(6):
        wall = Wall(130*i,240)
        MainGame.Wall_list.append(wall)

```

墙壁加入到窗口。

```

def blitWalls(self):
    for wall in MainGame.Wall_list:
        if wall.live:
            wall.displayWall()
        else:
            MainGame.Wall_list.remove(wall)

```

子弹不能穿墙

子弹类中新增方法，子弹与墙壁的碰撞，如果子弹与墙壁碰撞，修改子弹的状态，墙壁的生命值减少，如果墙壁的生命值小于等于零时候修改墙壁的状态。

```

#新增子弹与墙壁的碰撞
def hitWalls(self):
    for wall in MainGame.Wall_list:

```

```

if pygame.sprite.collide_rect(wall,self):
    #修改子弹的 live 属性
    self.live = False
    wall.hp -= 1
    if wall.hp <= 0:
        wall.live = False

```

坦克不能穿墙

如果坦克与墙壁碰撞，则坦克不能继续移动，需要修改坦克的坐标为移动之前的。因此在坦克类中新增属性 `oldLeft`、`oldTop` 记录移动之前的坐标，新增 `stay()`、`hitWalls()` 方法。

```

def stay(self):
    self.rect.left = self.oldLeft
    self.rect.top = self.oldTop
#新增碰撞墙壁的方法
def hitWalls(self):
    for wall in MainGame.Wall_list:
        if pygame.sprite.collide_rect(wall,self):
            self.stay()

```

双方坦克之间的碰撞检测

如果我方坦克碰撞到敌方坦克，则我方坦克再不能继续移动。同理如果敌方坦克碰撞到我方坦克也不能继续移动。

在我方坦克类中新增我方坦克与敌方坦克碰撞的方法。

```

class MyTank(Tank):
    def __init__(self,left,top):
        super(MyTank, self).__init__(left,top)
#新增主动碰撞到敌方坦克的方法
    def hitEnemyTank(self):
        for eTank in MainGame.EnemyTank_list:
            if pygame.sprite.collide_rect(eTank,self):
                self.stay()

```

我方坦克移动后，调用是否与敌方坦克发生碰撞。

```

#根据坦克的开关状态调用坦克的移动方法

```

```

if MainGame.TANK_P1 and not MainGame.TANK_P1.stop:
    MainGame.TANK_P1.move()
    #调用碰撞墙壁的方法
    MainGame.TANK_P1.hitWalls()
    MainGame.TANK_P1.hitEnemyTank()

```

在敌方坦克类中，新增敌方坦克碰撞我方坦克的方法。

```

def hitMyTank(self):
    if MainGame.TANK_P1 and MainGame.TANK_P1.live:
        if pygame.sprite.collide_rect(self, MainGame.TANK_P1):
            # 让敌方坦克停下来 stay()
            self.stay()

```

敌方坦克添加到窗口时候，调用是否与我方坦克碰撞。

```

#将敌方坦克加入到窗口中
def blitEnemyTank(self):
    for eTank in MainGame.EnemyTank_list:
        if eTank.live:
            eTank.displayTank()
            # 坦克移动的方法
            eTank.randMove()
            #调用敌方坦克与墙壁的碰撞方法
            eTank.hitWalls()
            #敌方坦克是否撞到我方坦克
            eTank.hitMyTank()
            # 调用敌方坦克的射击
            eBullet = eTank.shot()
            # 如果子弹为 None。不加入到列表
            if eBullet:
                # 将子弹存储敌方子弹列表
                MainGame.Enemy_bullet_list.append(eBullet)
        else:
            MainGame.EnemyTank_list.remove(eTank)

```


坦克大战之音效处理

music 是 pygame 中控制流音频的 pygame 模块，音乐模块与 pygame.mixer 紧密相连，pygame.mixer 是一个用来处理声音的模块，其含义为“混音器”。游戏中对声音的处理一般包括制造声音和播放声音两部分。使用 pygame.mixer.music.load()加载一个播放音乐的文件，pygame.mixer.music.play() 开始播放音乐流。

初始化音效类。

```
class Music():
    def __init__(self, fileName):
        self.fileName = fileName
        #先初始化混合器
        pygame.mixer.init()
        pygame.mixer.music.load(self.fileName)
    #开始播放音乐
    def play(self):
        pygame.mixer.music.play()
```

创建坦克时，添加音效。

```
#创建我方坦克的方法
def creatMyTank(self):
    # 创建我方坦克
    MainGame.TANK_P1 = MyTank(400, 300)
    #创建音乐对象
    music = Music('img/start.wav')
    #调用播放音乐方法
    music.play()
```

我方坦克发射子弹时，添加音效。

```
elif event.key == pygame.K_SPACE:
    print("发射子弹")
    if len(MainGame.Bullet_list) < 3:
        # 产生一颗子弹
        m = Bullet(MainGame.TANK_P1)
        # 将子弹加入到子弹列表
        MainGame.Bullet_list.append(m)
        music = Music('img/fire.wav')
```

```
music.play()
```