
目录

9.1 开发环境介绍.....	2
9.2 游戏的设计.....	5
9.2.1 界面的创建.....	6
9.2.2 类的实现.....	8
9.2.3 图像的显示.....	14
9.2.4 事件的处理.....	17
9.2.5 文字的绘制.....	20
9.3 五子棋游戏设计.....	23
9.3.1 五子棋简介.....	23
9.3.2 人人模式.....	24
9.3.3 人机模式.....	29
9.3.4 Alpha-Beta 剪枝搜索算法.....	33

第 9 章 设计有趣的游戏

本章将带领大家利用 Pygame 模块设计简单有趣的小游戏。Pygame 是一组用来开发游戏软件的 Python 程序模块，这个模块简单易用，很适合开发小游戏。本章首先介绍 Pygame 模块的安装及环境配置。接下来，以坦克大战的小游戏为例来介绍 Pygame 模块中常用函数的使用。此外，本章还介绍了五子棋游戏的实现，向大家展示了五子棋人人模式的设计，以及人机模式下电脑的落子决策。本章的重点在于让大家能够掌握利用 Pygame 设计小游戏，同时能够理解游戏中的头脑部分，也就是游戏的精华、思想。例如，理解本章中大家常玩的五子棋是如何设计出来的，电脑是如何下棋的这些思想。

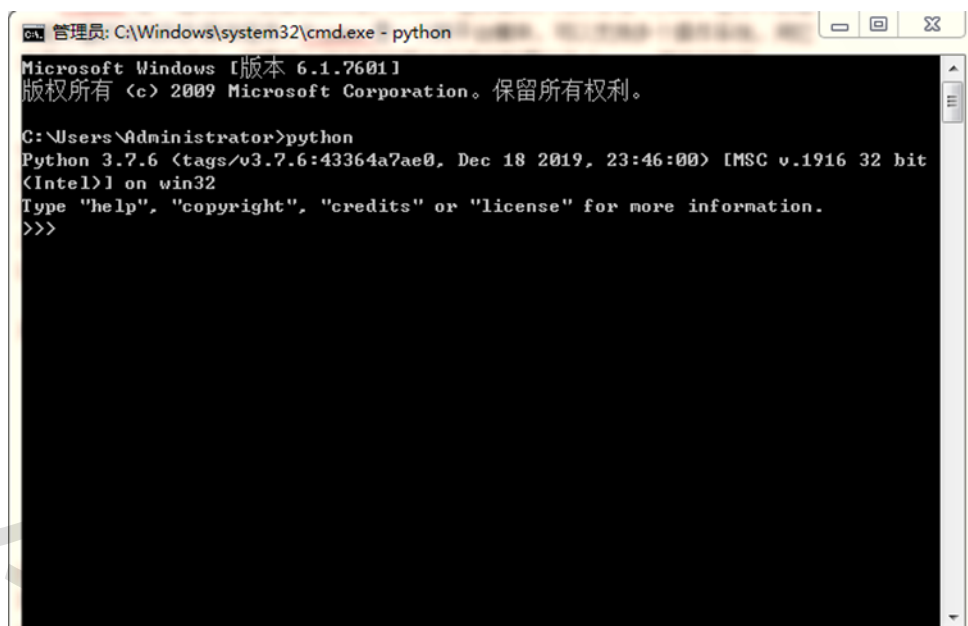
9.1 开发环境介绍

Pygame 是一组用来开发游戏软件的 Python 程序模块，允许你在 Python 程序中创建功能丰富的游戏和多媒体程序。而且 Pygame 是一个跨平台模块，可以支持多个操作系统，用它来开发小游戏非常适合。首先我们将介绍游戏的开发环境的配置以及 Pygame 模块的安装。

1. Python 的安装和配置

在浏览器中搜索“Python 下载”，打开官网找到需要下载的 Python 版本，点击下载并安装。安装完成后需要设置环境变量，做法是：【右键计算机】→【属性】→【高级系统设置】→【环境变量】，在第二个内容框“系统变量”中找到变量名为 path 的一行，双击打开，先在变量值最后添加一个分号“;”来作为与前面的间隔，然后再把 Python 的安装目录添加到变量值中，点击确定即可。例如，我将 python 安装到了 E 盘的 Python 文件夹下，所以我的 Python 安装目录是 E:\Python，将它添加到 path 的变量值中就可以了。

最后，需要检验一下 Python 是否安装配置成功。做法是：打开 cmd，输入 python，如果出现以下界面，则说明你的 Python 安装成功了，如下图所示。



```
管理员: C:\Windows\system32\cmd.exe - python
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>python
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 18 2019, 23:46:00) [MSC v.1916 32 bit
<Intel>] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

图 9-1 Python 安装测试图

2. Pygame 的安装

Pygame 是 Python 的一个库，安装 Pygame 库的过程与安装其他库的过程类似，都是通过 pip 来安装的。pip 是 Python 包管理工具，该工具提供了对 Python 包的查找、下载、安装、卸载的功能。Python 2.7.9 及后续版本，Python 3.4 及后续版本已经默认安装了 pip。如果不是必须使用某个较早的 Python 版本，建议在 Python 官网下载最新的 Python 版本。

为了便于安装 Pygame，需要再设置 pip 的环境变量，即是把 pip 的安装目录加入到系统变量 path 中，之后便可以通过 pip 来安装 Pygame。配置 pip 的环境变量的步骤和配置 Python 的环境变量类似，不同的是将 pip 的安装目录添加到系统变量中 path 的变量值中，pip 是在安装的 Python 里的 Scripts 文件夹下。例如，我将 python 安装到了 E 盘的 python 文件夹下，所以我的 pip 的安装目录是 E:\Python\Scripts。pip 的环境变量配置完成之后，关闭之前的 cmd 窗口，重新打开一个 cmd 窗口，输入 pip，如果出现以下界面，说明 pip 的环境变量配置成功，如下图所示。

```
管理员: C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>pip

Usage:
  pip <command> [options]

Commands:
  install           Install packages.
  download          Download packages.
  uninstall         Uninstall packages.
  freeze           Output installed packages in requirements format.
  list             List installed packages.
  show            Show information about installed packages.
  check           Verify installed packages have compatible dependencies.
  config          Manage local and global configuration.
  search          Search PyPI for packages.
  wheel          Build wheels from your requirements.
  hash           Compute hashes of package archives.
  completion     A helper command used for command completion.
  debug          Show information useful for debugging.
  help           Show help for commands.
```

图 9-2 pip 环境变量配置图

接下来，便可以用 pip 安装 Pygame 模块了。输入命令 `pip install pygame`，等待几秒之后便会自动下载，下载完成之后会自动安装，如下图所示。

```
管理员: C:\Windows\system32\cmd.exe
C:\Users\Administrator>pip install pygame
Collecting pygame
  Downloading https://files.pythonhosted.org/packages/80/2c/3a52e7e9c097229b026b4efbe6711c600f3a84ffdc5f11fd9e7f8932368e/pygame-1.9.6-cp37m-win32.whl (4.0MB)
|#####| 1.9MB 34kB/s eta 0:01:00
|#####| 1.9MB 34kB/s eta 0:01:00
|#####| 1.9MB 45kB/s eta 0:00:45
|#####| 1.9MB 45kB/s eta 0:00:45
|#####| 2.0MB 45kB/s eta 0:00:45
|#####| 2.0MB 45kB/s eta 0:00:45
|#####| 2.0MB 45kB/s eta 0:00:45
|#####| 2.0MB 45kB/s eta 0:00:45
|#####| 2.0MB 45kB/s eta 0:00:44
|#####| 2.0MB 45kB/s eta 0:00:44
|#####| 2.0MB 45kB/s eta 0:00:44
|#####| 2.0MB 45kB/s eta 0:00:44
|#####| 2.0MB 45kB/s eta 0:00:43
|#####| 2.0MB 84kB/s eta 0:00:2
|#####| 2.0MB 84kB/s eta 0:00:2
|#####| 2.1MB 84kB/s eta 0:00:2
|#####| 2.1MB 84kB/s eta 0:00:2
|#####| 2.1MB 84kB/s eta 0:00:2
|#####| 2.1MB 84kB/s eta 0:00:2
|#####| 2.1MB 84kB/s eta 0:00:2
```

图 9-3 Pygame 下载图

3. PyCharm 中 Pygame 的配置

PyCharm 是一种 Python IDE，带有一整套工具，可以帮助用户在使用 Python 语言开发时提高效率，比如调试、语法高亮、Project 管理、代码跳转、智能提示、自动完成、单元测试、版本控制等工具。本节选择 PyCharm 作为开发工具，学生可在 Pycharm 官网下载此软件。安装此软件之后，要想使用 PyCharm 运行小游戏，还需要在 PyCharm 中配置 Python 解释器。

打开 PyCharm 软件，然后点击【File】→【Settings】→【Project Interpreter】，便进入 Project Interpreter 界面，如下图所示。点击“Project Interpreter”右侧的小

三角下拉框，点击该按钮之后，Pycharm 会自动弹出 Python 的解释器路径，选择自己安装的 Python 解释器，Python 解释器就配置成功了。

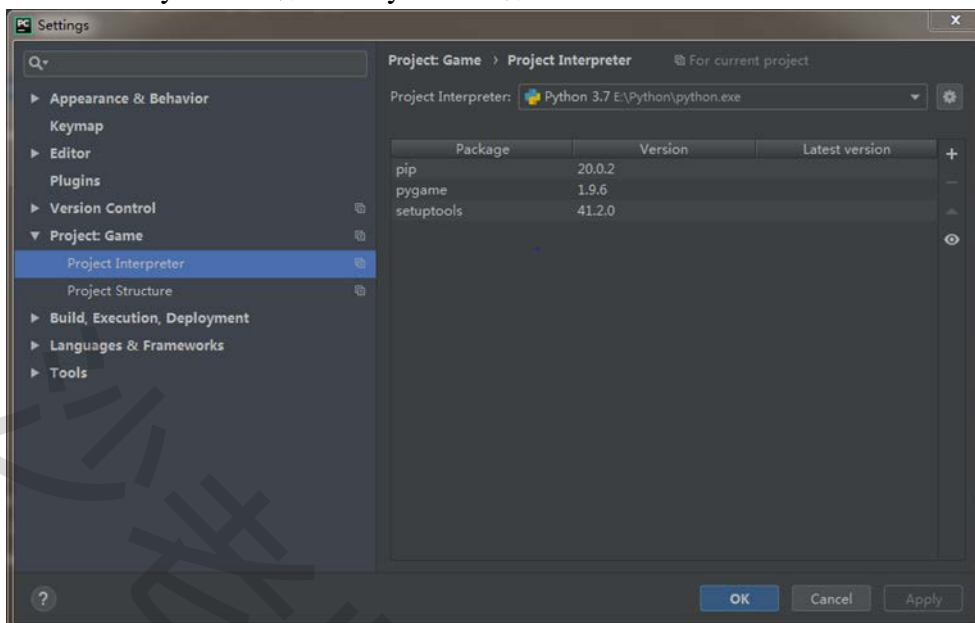


图 9-4 Python 解释器配置图

9.2 游戏的设计

上一小节我们已经介绍了 Pygame 开发游戏的环境配置，接下来将介绍如何使用 Pygame 写游戏。Pygame 常用模块概览如下表所示。更详细的模块介绍请参考 Pygame 官网 <https://www.pygame.org/docs/ref/event.html>。

表 9-1 pygame 常用模块概览表

模块名	功能
pygame.display	控制显示窗口和屏幕
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	加载和播放声音
pygame.mouse	使用鼠标
pygame.music	播放音频
pygame.Color	用来描述颜色

本节以“坦克大战”小游戏为例来介绍如何使用 pygame 设计游戏。在小游戏坦克大战中，游戏的主要对象为我方坦克，敌方坦克，两方子弹。玩家可以通过键盘的上下左右键来控制坦克的移动，空格键控制发射子弹。玩家在躲避敌方坦克的同时，通过发射子弹来消灭敌方坦克，当敌方坦克全部被消灭，就是通关成

功。如若玩家阵亡，按 Esc 键可以复活。此游戏的设计包含 6 个类，游戏类 `MainGame()`，坦克类 `Tank()`，我方坦克类 `MyTank()`，敌方坦克类 `EnemyTank()`，子弹类 `Bullet()`，爆炸类 `Explode()`，我们将一一介绍。

9.2.1 界面的创建

在使用 `pygame` 模块之前，需要做一些初始化工作，如导入 `pygame` 库、导入 `pygame` 中所有常量和 `sys` 中 `exit()` 函数等，若还需要其他模块可按需导入。

```
import pygame
from pygame.locals import *      #导入 pygame 中所有常量
from sys import exit            #需要 sys 中 exit() 函数，用来退出程序
```

接下来将介绍如何绘制游戏主界面。首先需要先绘制一个游戏窗口，并修改窗口的标题为游戏的名称，之后便可以在窗口内画出敌方坦克、我方坦克，以及绘制文字。`pygame` 中用 `pygame.display` 模块来控制窗口和屏幕显示。这个模块提供了控制 `pygame` 显示界面 (`display`) 的各种函数。此模块有以下几个常用函数：

- `pygame.display.init()` — 初始化 `display` 模块
- `pygame.display.set_mode()` — 创建窗口
- `pygame.display.set_caption()` — 设置当前窗口的标题
- `pygame.display.update()` — 显示图像
- `pygame.display.flip()` — 显示图像

`pygame.display.init()` 函数的作用是初始化 `display` 模块，在使用 `display` 模块之前都需要先调用此函数进行初始化操作。

`pygame.display.set_mode(resolution=(0,0),flags=0,depth=0)` 函数的作用是创建一个窗口，返回一个特定大小和属性的 `Surface` 对象。在这三个参数中最常用的是 `resolution`，它用来控制生成窗口的大小，`flags` 代表的是扩展选项，`depath` 不推荐设置。在 `Pygame` 中窗口和图片都称为 `Surface`，所谓 `Surface` 对象在 `Pygame` 中就是用来表示图像的对象，图片是由像素组成的，`Surface` 对象具有固定的分辨率和像素格式。

`pygame.display.set_caption(title)` 函数的功能是将窗口的名称设置为“`title`”，其中参数 `title` 为字符串。若不设置游戏窗口的名称，窗口标题会显示为默认的“`pygame window`”。

`pygame.display.update()` 和 `pygame.display.flip()` 函数都可以用来将待显示的内容更新到屏幕上。不用的是，`pygame.display.flip()` 将整个屏幕待显示的内容显示到屏幕上，而 `pygame.display.update()` 是将部分内容显示到屏幕上，如果不指定参数，则与 `flip` 功能相同。

在坦克大战游戏中，首先使用 `pygame.display.set_mode()` 函数创建了一个分

分辨率为 1000*500 的窗口，并创建名称为 `window` 的窗口对象，以便后续对该窗口的操作。然后使用 `set_caption()`函数设置窗口的名称，将游戏窗口的名称设置为“坦克大战 1.01”。创建游戏窗口的代码如下。

```
#创建游戏窗口
import pygame
from pygame.locals import *      #导入 pygame 中所有常量
from sys import exit

SCREEN_WIDTH=1000                #设置窗口的宽度
SCREEN_HEIGHT=500                #设置窗口的高度

pygame.display.init()           #初始化 display 模块
window=pygame.display.set_mode([SCREEN_WIDTH, SCREEN_HEIGHT])
pygame.display.set_caption('坦克大战 1.01') #设置窗口名称
```

以上程序可以简单地创建一个游戏窗口，但是当运行程序时，会发现窗口打开后立刻就关闭了，这是因为程序已经运行结束了。一个游戏窗口应该是不断循环，直到玩家关闭窗口。解决这个问题的方法是添加循环语句，并检测事件，只有当玩家关闭窗口时，才退出游戏。具体代码如下：

```
import pygame
from pygame.locals import *
from sys import exit

SCREEN_WIDTH=1000
SCREEN_HEIGHT=500
BG_COLOR=pygame.Color(0, 0, 0)    #设置背景颜色为黑色

pygame.display.init()
window=pygame.display.set_mode([SCREEN_WIDTH, SCREEN_HEIGHT])
pygame.display.set_caption('坦克大战 1.01')

while True:
    window.fill(BG_COLOR)          # 将窗口填充为黑色
    pygame.display.flip()
    eventList = pygame.event.get() #获取事件存放到 eventList 中
    for event in eventList:
        if event.type == pygame.QUIT:
            print('谢谢使用，欢迎再次使用')
            exit() # 结束游戏
```

当添加好循环结构后，再次运行程序，可以发现弹出一个黑色的窗口并且窗口不会闪退，当点击右上角关闭窗口时，游戏窗口才会被关闭。`pygame.event.get()`是一个事件获取函数，关于游戏事件的处理后面将会介绍。

在程序中使用到了 `pygame.Color()`函数，它是用于表示颜色的 `pygame` 对象。通过调用 `pygame.Color(r, g, b,a)`函数来创建 `Color` 对象,返回的是一个表示颜色的

四元组，可以将此存储在变量中。`pygame.surface.fill(color)`对 `surface` 对象填充某一种颜色，主要是对背景实现填充，这里的 `surface` 是所有 `surface` 对象的统称，例如上面代码中的我们创建的 `window`。

坦克大战的游戏的设计是利用面向对象编程的方式，我们设计了一个游戏类 `MainGame()`，并创建了 `startGame()`方法，此方法用来控制坦克大战游戏的主要逻辑。我们将上述创建窗口的代码放在游戏类 `MainGame()`中，通过在程序主函数中调用游戏类的 `startGame()`方法来运行游戏。下面给出坦克大战中游戏类 `MainGame()`的部分代码。

```
import pygame
from pygame.locals import *
from sys import exit

SCREEN_WIDTH=1000
SCREEN_HEIGHT=500
BG_COLOR=pygame.Color(0,0,0)
#游戏类
class MainGame():
    window=None
    def __init__(self):
        pass
    def startGame(self):
        pygame.display.init()
        MainGame.window=pygame.display.set_mode([SCREEN_WIDTH,SCREEN_HEIGHT])
        pygame.display.set_caption('坦克大战 1.01')
        while True:
            MainGame.window.fill(BG_COLOR)
            pygame.display.flip()
            eventList = pygame.event.get()
            for event in eventList:
                if event.type == pygame.QUIT:
                    print('谢谢使用，欢迎再次使用')
                    exit()

if __name__ == '__main__':
    MainGame().startGame()
```

9.2.2 类的实现

1、坦克类的实现

在坦克大战游戏中，游戏的主要对象为我方坦克，敌方坦克，我方子弹以及敌方子弹。我方坦克、敌方坦克都有共同的坦克属性，所以我们先创建了一个坦克类 `Tank()`，然后敌方坦克 `MyTank()`和我方坦克类 `EnemyTank()`都继承此类。定义坦克类 `Tank()`时，首先定义坦克共有的一些初始化属性和方法，例如坦克的方

向，坦克移动的速度等。Tank()实现代码如下所示。

```
class Tank():
    def __init__(self, left, top):    #初始化坦克
        self.images={
            'U': pygame.image.load('p2tankU.png').convert(),
            'D': pygame.image.load('p2tankD.png').convert(),
            'L': pygame.image.load('p2tankL.png').convert(),
            'R': pygame.image.load('p2tankR.png').convert()
        }
        self.direction='L'    #设定坦克初始朝向为左
        self.image = self.images[self.direction]
        self.rect = self.image.get_rect()    #获得 image 对象的 rect
        self.rect.left=left
        self.rect.top=top
        self.speed = 5    #坦克移动的速度
        self.stop=True    #控制坦克移动
        self.live=True
        #保持原来的位置
        self.oldLeft = self.rect.left
        self.oldTop = self.rect.top
    # 坦克移动
    def move(self):
        self.oldLeft = self.rect.left    #保持原来的状态
        self.oldTop = self.rect.top
        #判断坦克的方向进行移动
        if self.direction == 'L':
            if self.rect.left>0:
                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<SCREEN_HEIGHT:
                self.rect.top +=self.speed
        elif self.direction == 'R':
            if self.rect.left+self.rect.height<SCREEN_WIDTH:
                self.rect.left += self.speed
    def shot(self):
        return Bullet(self)    #创建子弹对象
    def stay(self):    #保持原来的位置
        self.rect.left = self.oldLeft
        self.rect.top = self.oldTop
    def displayTank(self):    #画出坦克
        self.image = self.images[self.direction]
        MainGame.window.blit(self.image, self.rect)
```

我们将对以上代码使用到的 pygame 模块进行介绍。

(1) `img = pygame.image.load(filename)` 表示读取文件名为 filename 的图像

文件，一般来说支持 JPG、PNG、GIF 等多种图片类型。此函数返回一个包含图像的 Surface 对象。convert()函数对载入的图片做了转换，是一种优化处理，否则每次显示的时候，系统都需要转换一次。Tank()类通过 pygame.image.load 读取我方坦克 4 个图像，并存放到字典中，以便后续使用。

(2) image.get_rect()函数获取 image 对象的矩形区域 rect，返回一个包含整个 Surface 的新矩形，这个矩形的位置总是从窗口的(0,0)开始，宽度和高度与图像的大小相同。rect 包含 rect.top、rect.bottom、rect.right、rect.left，rect.center 属性，分别表示矩形的上、下、左、右、中心位置，可以通过给 rect 的属性赋值来指定图像显示的位置。

(3) blit(source, dest, area=None, special_flags=0)函数的功能是将图片绘制到屏幕相应坐标上。source 是待显示的图片对象，dest 是待显示图片的位置，后面两个参数默认，可以不传。dest 可以是代表左上角的一对坐标，也可以是 rect。当传入的 dest 是 rect 时，将会把 rect 的 top 和 left 用作 blit 的位置。调用 blit()这个动作的必须是一个 Surface 类的实例，比如代码中的 window 就是一个 Surface 实例对象。blit 的作用就是把 source 这个 Surface 对象画在 window 这个 Surface 对象上面。

坦克可以上下左右移动，根据坦克的朝向，可形成 4 种形态，我们把坦克的 4 种图片加载到 image 字典中，“U”、“D”、“L”、“R”分别对应坦克的朝向为上，下，左，右。self.direction='L'先初始化坦克的朝向为左，也就是游戏刚开始时坦克的朝向。然后我们获取相应 image 的矩形对象 rect，利用 rect 的属性 rec.top 和 rec.left 来设置坦克距离屏幕左上角的初始位置。接着我们定义了坦克类的一些方法，move()方法来表示坦克的移动，shot()方法来表示坦克发射子弹，其中 Bullet()是子弹类，将在后面介绍。

坦克类创建之后，我方坦克和敌方坦克就可以继承坦克类。我方坦克和敌方坦克类的代码如下。

```
#我方坦克类
class MyTank(Tank):          #继承坦克类
    def __init__(self, left, top):
        super(MyTank, self).__init__(left, top) #初始化我方坦克的位置
    #检查我方坦克与敌方坦克发生碰撞
    def myTank_hit_enemyTank(self):
        for enemyTank in MainGame.enemyTankList:
            if pygame.sprite.collide_rect(self, enemyTank):
                self.stay() #停止当前方向的运动，而不是互相穿过
#敌方坦克类
class EnemyTank(Tank):     #继承坦克类
    def __init__(self, left, top, speed): #初始化敌方坦克类
        super(EnemyTank, self).__init__(left, top)
```

```

self.images={
    'U': pygame.image.load('p3tankU.png').convert(),
    'D': pygame.image.load('p3tankD.png').convert(),
    'L': pygame.image.load('p3tankL.png').convert(),
    'R': pygame.image.load('p3tankR.png').convert()
} #加载敌方坦克的四种图片

self.direction = self.randDirection() #随机生成敌方坦克方向
self.image = self.images[self.direction]
self.rect = self.image.get_rect()
self.rect.left=left
self.rect.top=top
self.speed=speed
self.step=60
#检测敌方坦克与我方坦克的碰撞
def enemyTank_hit_myTank(self):
    if pygame.sprite.collide_rect(self, MainGame.my_tank):
        self.stay() #停止当前方向的运动，而不是互相穿过
#随机产生一个方向
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'
def randMove(self):
    if self.step<=0:
        self.step=60
        self.direction = self.randDirection()
    else:
        self.move()
        self.step-=1 #坦克移动的速度递减
def shot(self):
    num = random.randint(1,100)
    if num<3:
        return Bullet(self)

```

以上代码出现了一个新的函数 `pygame.sprite.collide_rect(left,right)`，此函数是使用 `rect` 属性进行精灵（`sprite`）间的冲突检测，就是测试两个图像之间是否发生了碰撞，函数返回值是 `False` 或 `True`。`True` 表示精灵之间产生了碰撞，`False` 表示没有产生碰撞。

2、子弹类和爆炸对象类的实现

前面已经介绍了坦克类、我方坦克类以及敌方坦克类的实现，接下来我们介绍一下子弹类以及爆炸对象类的设计。

```

#子弹类
class Bullet():
    def __init__(self, tank):
        self.image = pygame.image.load('steels.png').convert()
        #坦克的方向决定子弹的方向
        self.direction = tank.direction
        self.rect = self.image.get_rect()
        #根据子弹射出的方向来设定子弹射出的左上角的位置
        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
        self.speed=6 #子弹的速度
        self.live=True #子弹的状态
    def move(self):
        if self.direction == 'U':
            if self.rect.top>0:
                self.rect.top-=self.speed
            else:
                self.live=False
        elif self.direction == 'R':
            if self.rect.left+self.rect.width<SCREEN_WIDTH:
                self.rect.left+=self.speed
            else:
                self.live=False
        elif self.direction == 'D':
            if self.rect.top+self.rect.height<SCREEN_HEIGHT:
                self.rect.top+=self.speed
            else:
                self.live=False
        elif self.direction == 'L':
            if self.rect.left>0:
                self.rect.left-=self.speed
            else:
                self.live=False
#我方子弹和敌方坦克的碰撞

```

```

def myBullet_hit_enemyTank(self):
    for enemyTank in MainGame.enemyTankList:
        if pygame.sprite.collide_rect(enemyTank, self):
            enemyTank.live = False #修改敌方坦克的状态
            self.live = False #修改子弹的状态
            explode = Explode(enemyTank) #产生一个爆炸队对象
            MainGame.explodeList.append(explode)
#画出子弹
def displayBullet(self):
    MainGame.window.blit(self.image, self.rect)
#敌方子弹与我方坦克的碰撞
def enemyBullet_hit_myTank(self):
    if MainGame.my_tank and MainGame.my_tank.live:
        if pygame.sprite.collide_rect(MainGame.my_tank, self):
            explode = Explode(MainGame.my_tank)
            MainGame.explodeList.append(explode)
            self.live=False
            MainGame.my_tank.live=False

```

实现子弹类时，首先初始化子弹的方向、发射子弹的位置、子弹的速度和状态。move()方法是根据子弹的方向和速度来计算子弹的位置，实现子弹的运动，displayBullet()方法实现把子弹画在屏幕上。enemyBullet_hit_myTank()和myBullet_hit_enemyTank()是为了检测子弹与坦克的碰撞，当敌方子弹与我方坦克碰撞以及我方子弹与敌方坦克碰撞时会产生爆炸现象，因此创建了爆炸类Explode()，并修改坦克和子弹的状态。下面我们给出爆炸类的代码实现。

```

#爆炸类
class Explode():
    def __init__(self, tank):
        #爆炸的位置有当前子弹打中的位置确定
        self.rect=tank.rect
        self.images=[
            pygame.image.load('blast1.gif').convert(),
            pygame.image.load('blast2.gif').convert(),
            pygame.image.load('blast3.gif').convert(),
            pygame.image.load('blast4.gif').convert(),
            pygame.image.load('blast5.gif').convert(),
        ]
        self.step=0
        self.image = self.images[self.step]
        self.live=True
#显示爆炸效果
def displayExplode(self):
    #根据索引获取爆炸对象
    if self.step < len(self.images):
        self.image = self.images[self.step]
        self.step+=1
        MainGame.window.blit(self.image, self.rect)

```

```
else:
    self.live=False
    self.step=0
```

9.2.3 图像的显示

前面小节部分已经实现了我方坦克类、敌方坦克类、子弹类、爆炸类，接下来将介绍利用类来实例化对象，并要把这些对象画在屏幕上。游戏图像的绘制都是在游戏类中 `MainGame()`中实现的，首先，我们需要利用实现的坦克类来创建出我方坦克和敌方坦克对象，相关代码如游戏类所示。

```
#游戏类
class MainGame():
    window=None
    my_tank = None          #我方坦克
    enemyTankCount = 6     #敌军坦克的数量
    enemyTankList=[]       #存储敌方坦克的列表
    myBulletList = []      #存储我方坦克子弹的列表
    enemyBulletList=[]     #存储敌方坦克子弹的列表
    explodelist = []      #爆炸列表
    def __init__(self):
        pass
    def startGame(self):
        pygame.display.init()
        MainGame.window=pygame.display.set_mode([SCREEN_WIDTH, SCREEN_HEIGHT])
        pygame.display.set_caption('坦克大战 1.01')
        self.createMyTank()    #创建我方坦克
        self.createEnemyTank() #创建敌方坦克
        while True:
            time.sleep(0.02)
            MainGame.window.fill(BG_COLOR)
            eventList = pygame.event.get()
            for event in eventList:
                if event.type == pygame.QUIT:
                    print('谢谢使用，欢迎再次使用')
                    exit() # 结束游戏

#创建我方坦克对象
def createMyTank(self):
    MainGame.my_tank = MyTank(350, 300)

#创建 6 个敌方坦克对象
def createEnemyTank(self):
    top=100
    for i in range(MainGame.enemyTankCount):
        left = random.randint(68, 500)
        speed = random.randint(1, 4)
        enemy=EnemyTank(left, top, speed)
        MainGame.enemyTankList.append(enemy)
```

在游戏类代码中，`createMyTank()`和 `createEnemyTank()`函数的目的是创建我

方坦克和敌方坦克对象。`createEnemyTank()`函数先随机产生坦克的速度、位置以及方向，然后循环创建 `enemyTankCount` 个敌方坦克对象，并初始化每个敌方坦克对象的位置、速度、方向等属性，然后将每个坦克对象存储到 `enemyTankList` 列表中，方便后续操作。这样就创建了 `enemyTankCount` 个具有不同位置、不同方向、不同速度的坦克对象。`createMyTank()`函数用来创建我方坦克对象，创建我方坦克对象时需要传入坦克的初始位置。

目前已经实现了坦克对象的创建，如何把这些坦克对象，也就是所谓的图像显示在屏幕上呢？将坦克对象、子弹对象以及爆炸对象画在屏幕上，也就是将一个 `Surface` 对象画在另外一个 `Surface` 对象上，`blit` 函数可以实现这个功能。画坦克时，需要先判断坦克的状态，若坦克依旧存活，就调用坦克类中的 `display` 函数把坦克画在屏幕上，`display` 函数也就是利用 `blit` 函数画坦克的封装，具体代码可参见坦克类。代码如下。

```
class MainGame():
    window=None
    my_tank = None
    enemyTankCount = 6          #敌军坦克的数量
    enemyTankList=[]          #存储敌方坦克的列表
    myBulletList = []          #创建存储我方坦克子弹的列表
    enemyBulletList=[]        #创建存储敌方子弹的列表
    explodelist = []          #创建爆炸列表
    def __init__(self):
        pass
    def startGame(self):
        pygame.display.init()
        MainGame.window=pygame.display.set_mode([SCREEN_WIDTH, SCREEN_HEIGHT])
        pygame.display.set_caption('坦克大战 1.01')
        self.createMyTank()
        self.createEnemyTank()
        while True:
            time.sleep(0.02)
            MainGame.window.fill(BG_COLOR)
            eventList = pygame.event.get()
            for event in eventList:
                if event.type == pygame.QUIT:
                    print('谢谢使用，欢迎再次使用')
                    exit() # 结束游戏
            self.blitMyTank()      #画出我方坦克
            self.blitEnemyTank()  #画出敌方坦克
            self.blitEnemyBullet() #画出敌方坦克的子弹
            self.blitMyBullet()   #画出我方坦克的子弹
            self.blitExplode()    #展示爆炸现象
            pygame.display.update() #将所画图形显示到屏幕上

    def createMyTank(self):
```

```

    MainGame.my_tank = MyTank(350, 300)
def createEnemyTank(self):
    top=100
    for i in range(MainGame.enemyTankCount):
        left = random.randint(68, 500)
        speed = random.randint(1, 4)
        enemy=EnemyTank(left, top, speed)
        MainGame.enemyTankList.append(enemy)
#画出我方坦克
def blitMyTank(self):
    if MainGame.my_tank and MainGame.my_tank.live:
        MainGame.my_tank.displayTank() # 画出我方坦克
    else:
        del MainGame.my_tank # 删除 my_tank 实例对象
        MainGame.my_tank = None
#画出敌方坦克
def blitEnemyTank(self):
    for enemyTank in MainGame.enemyTankList:
        if enemyTank.live:
            EnemyTank.displayTank(enemyTank)
            enemyTank.randMove()
            if MainGame.my_tank and MainGame.my_tank.live:
                enemyTank.enemyTank_hit_myTank()
                enemyBullet=enemyTank.shot() #创建敌军子弹对象, 放到列表
                if enemyBullet:
                    MainGame.enemyBulletList.append(enemyBullet)
            else: #被消灭 删除
                MainGame.enemyTankList.remove(enemyTank)
#画出我方坦克子弹
def blitMyBullet(self):
    for myBullet in MainGame.myBulletList:
        if myBullet.live:
            myBullet.displayBullet()
            myBullet.move()
            myBullet.myBullet_hit_enemyTank()#检测碰撞
        else:
            MainGame.myBulletList.remove(myBullet)
#画出敌军坦克的子弹
def blitEnemyBullet(self):
    for enemyBullet in MainGame.enemyBulletList:
        if enemyBullet.live:
            enemyBullet.displayBullet()
            enemyBullet.move()
            enemyBullet.enemyBullet_hit_myTank()
        else:
            MainGame.enemyBulletList.remove(enemyBullet)
#画出爆炸对象
def blitExplode(self):
    for explode in MainGame.explodeList:

```



```

if explode.live:
    explode.displayExplode()
else:
    MainGame.explodeList.remove(explode)

```

我们在游戏类 `MainGame` 中定义并调用了五种方法：`blitMyTank()`、`blitEnemyTank()`、`blitMyBullet()`、`blitEnemyBullet()`、`blitExplode()`分别来画出我方坦克、敌方坦克、我方子弹、敌方子弹以及爆炸效果。在调用了以上 5 种方法之后，必须要使用函数 `pygame.display.update()`将所画图形显示到屏幕上，否则屏幕上不会出现图形。

游戏进行到这里已经基本实现了坦克随机移动、子弹发射、爆炸效果等效果，但是目前我方坦克还不能移动，只能在初始位置发射子弹。由于我方坦克的运动是由玩家通过键盘控制的，所以我们需要添加对键盘事件的处理，来使得玩家能够通过键盘控制我方坦克的移动。

9.2.4 事件的处理

事件是一个概念，比如点击鼠标左键，按下一个键盘的按键，关闭窗口等等都是一个事件。`pygame` 提供了一个 `pygame.event.get()`函数来获取游戏的事件，并把它们存放在一个队列中，程序通过读取整个事件队列，来获取当前发生的事件，并作出响应。常见的有以下几种事件，如下表所示。

表 9-2 常用事件表

事件	产生途径	参数
QUIT	用户点击关闭按钮	none
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

坦克大战游戏中涉及到的事件有游戏退出事件、键盘事件。我们需要使用键盘的上下左右键来控制我方坦克的移动，用 `SPACE` 键来控制我方坦克发射子弹。

1、处理键盘事件

处理事件时，首先遍历事件列表，判断事件的类型，然后根据不同的事件类型做出不同的响应。键盘事件包括键盘按下和键盘释放，也就是对应 `pygame.KEYDOWN` 和 `pygame.KEYUP` 这两个事件。键盘按下事件的判断也就是判断事件类型是否是 `pygame.KEYDOWN`，如果是，再进一步判断键值是否是我们需要的那个键位，然后执行对应的操作。判断事件类型用 `event.type`，判断键值用 `event.key`。键值 `event.key` 是一个数字，`Pygame` 中用 `K_xxx` 来表示。例如，

`pygame.K_w`，表示按键 `w`，`pygame.K_ESCAPE`，则表示按键 `Esc`。

以下是一个键盘事件小案例，主要是为了让同学们能够更好的理解以及使用事件处理的函数。此案例实现的是按下“`W`”键（也可以设置成其他键）屏幕上显示一只小狗的图片，也可以显示其他图片，只需要更换一下文件名，并把图片放到当前文件夹即可。

```
#键盘事件小案例
import pygame
from pygame.locals import *
from sys import exit

pygame.init()
screen = pygame.display.set_mode((500, 800))
pygame.display.set_caption("键盘事件小案例")
while True:
    for event in pygame.event.get():
        if event.type == QUIT: #游戏退出事件
            exit()
        if event.type == KEYDOWN: #键盘按下事件
            if event.key == pygame.K_w: #按下 W 键时在屏幕上显示出小狗的图片
                print("The button is pressed")
                image = pygame.image.load("dog.jpg").convert()
                rect = image.get_rect()
                screen.blit(image, rect)
                pygame.display.update()
```

2、处理鼠标事件

鼠标事件有按键单击，包括左键，滑轮，中键。`pygame.event.get()`函数还会给我们返回鼠标的坐标(`pos`)。我们给出了一个鼠标事件小案例，在这个程序中，我们判断是否有鼠标按下事件，如果有按键，就打印出鼠标的坐标值，这个值是相对窗口来说的，也就是你的鼠标点击窗口的左上顶点处，`event.pos` 将返回(0,0)。

```
#鼠标事件小案例
import pygame
from pygame.locals import *
from sys import exit

pygame.init()
screen = pygame.display.set_mode((600, 500))
pygame.display.set_caption("鼠标事件小案例")

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
        if event.type == MOUSEBUTTONDOWN:
            mouse_x, mouse_y = event.pos
```

```
print(mouse_x, mouse_y)
```

接下来我们看一下坦克大战中对于键盘事件的处理。我们将所有的事件处理写到一个函数 `getEvent()` 中，将 `getEvent()` 放入游戏类 `MainGame` 中，并在游戏循环中调用，持续获取事件。一个游戏循环（也可以称为主循环）就做这 3 件事：处理事件，更新游戏状态，绘制游戏状态到屏幕上。由于代码过长，只展示 `MainGame` 类中 `getEvent()` 的代码，其他方法在上面已经展示过。代码如下。

```
#事件处理
class MainGame():
    window=None
    my_tank = None
    enemyTankCount = 6
    enemyTankList=[]
    myBulletList = []
    enemyBulletList=[]
    explodeList = []
    def __init__(self):
        pass
    def startGame(self):
        pygame.display.init()
        MainGame.window=pygame.display.set_mode([SCREEN_WIDTH, SCREEN_HEIGHT])
        pygame.display.set_caption('坦克大战 1.01')
        self.createMyTank()
        self.createEnemyTank()
        while True:
            time.sleep(0.02)
            MainGame.window.fill(BG_COLOR)
            self.getEvent() #调用事件函数
            self.blitMyTank()
            self.blitEnemyTank()
            self.blitEnemyBullet()
            self.blitMyBullet()
            self.blitExplode()
            if MainGame.my_tank and MainGame.my_tank.live:
                if not MainGame.my_tank.stop:#坦克可以移动
                    MainGame.my_tank.move()
                    MainGame.my_tank.myTank_hit_enemyTank()
            pygame.display.update()
            ..... (此处省略之前展示过的代码)
            .....
#事件处理函数
def getEvent(self):
    eventList=pygame.event.get() #获取事件
    for event in eventList:
        if event.type == pygame.QUIT:
            print('谢谢使用，欢迎再次使用') #结束游戏
            exit()
```

```

if event.type == pygame.KEYDOWN:    #键盘事件
    if len(MainGame.enemyTankList) == 0:
        if event.key==pygame.K_ESCAPE:
            self.createEnemyTank()
    if not MainGame.my_tank:
        if event.key== pygame.K_ESCAPE:
            self.createMyTank()
    if MainGame.my_tank and MainGame.my_tank.live:
        #判断上下左右
        if event.key == pygame.K_LEFT:
            MainGame.my_tank.direction='L'
            MainGame.my_tank.stop=False
            print('按下左键, 坦克向左移动')
        elif event.key == pygame.K_RIGHT:
            MainGame.my_tank.direction='R'
            MainGame.my_tank.stop = False
            print('按下右键, 坦克向右移动')
        elif event.key == pygame.K_UP:
            MainGame.my_tank.direction='U'
            MainGame.my_tank.stop = False
            print('按下上键, 坦克向上移动')
        elif event.key == pygame.K_DOWN:
            MainGame.my_tank.direction='D'
            MainGame.my_tank.stop = False
            print('按下下键, 坦克向下移动')
        elif event.key == pygame.K_SPACE:
            print('发射子弹')
            if len(MainGame.myBulletList)<3:
                myBullet=Bullet(MainGame.my_tank)
                MainGame.myBulletList.append(myBullet)
    if event.type == pygame.KEYUP: #释放键盘事件
        if MainGame.my_tank and MainGame.my_tank.live:
            #判断释放键是上下左右键才停止
            if event.key == pygame.K_UP or event.key == pygame.K_DOWN or
event.key == pygame.K_LEFT or event.key == pygame.K_RIGHT:
                MainGame.my_tank.stop = True

```

添加了事件处理函数之后，我们便可以通过键盘控制坦克移动了。此时坦克大战的小游戏基本完成了。

9.2.5 文字的绘制

接下来我们讲一下文字的绘制。游戏的设计往往少不了文字的说明，那么如何在屏幕上展现出我们想要的文字呢？首先看一个绘制文字的小案例。

```

#绘制文字小案例
import pygame
from pygame.locals import *
from sys import exit
TEXT_COLOR = pygame.Color(255, 255, 255) #定义字体的颜色为白色

```

```

pygame.init()
screen = pygame.display.set_mode((500, 300))
pygame.display.set_caption("绘制字体")
font=pygame.font.SysFont('kaiti', 24) #创建一个字体对象
#创建文本 Surface 对象
textSurface=font.render("I Love Python", True, TEXT_COLOR)
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
        screen.blit(textSurface, (150, 150)) #将文本 Surface 对象画到屏幕上
    pygame.display.update()

```

在绘制文字小案例中，主要实现的是在指定的位置(150,150)处绘制文本内容“ I Lvoe Python”，并且将字体的颜色设置为白色。绘制文字时，首先通过 `pygame.font.SysFont(name, size, bold=False, italic=False)` 函数从系统字体库创建一个 `Font` 对象，并返回从系统字体中加载的新字体对象，其中参数 `name` 是字体类型，`size` 是字体大小。`render(text, antialias, color, background=None)` 函数创建一个新的 `Surface` 对象，并在 `Surface` 对象上渲染指定的文本，其中第一个参数表示字体的类型，第二个参数表示字体大小。`Pygame` 没有提供直接的方式在一个现有的 `Surface` 对象上绘制文本，取而代之的方法是，使用 `render()` 函数创建一个渲染了文本的图像（`Surface` 对象），然后将这个图像绘制到目标 `Surface` 对象上，并且仅支持渲染一行文本：“换行”字符不会被渲染。

接下来我们看一下如何在坦克大战游戏中添加字体。坦克大战游戏可以用 `ESC` 键复活或者重新开始游戏，可以用 `SPACE` 键发射子弹，可以用方向键控制坦克运动，这些说明只有设计游戏的人知道，玩家却不知道，所以我们需要把这些文字绘制在屏幕上，让玩家一看便知游戏规则。我们在游戏类 `MainGame` 中设计了一个方法 `getTextSurface()` 用于获取文本对象，然后在游戏主循环中使用 `blit` 函数将文本对象画在屏幕上的指定位置处。除此之外，当我方坦克被歼灭时，我们也会在屏幕上显示“您的坦克已被敌方消灭，是否复活？”的文字提示，当敌军坦克全部被歼灭是，也会出现“游戏胜利，是否重新开始？”的文字提示，所以这些文字提示也需要显示在屏幕上。具体实现代码如下。

```

#坦克大战文字的绘制
SCREEN_WIDTH=1000
SCREEN_HEIGHT=500
BG_COLOR=pygame.Color(0, 0, 0)
TEXT_COLOR=pygame.Color(255, 255, 255) #设置文字的颜色

class MainGame():
    window=None
    my_tank = None

```

```

enemyTankCount = 6
enemyTankList=[]
myBulletList = []
enemyBulletList=[]
explodeList = []
def __init__(self):
    pass
def startGame(self):
    pygame.display.init()
    MainGame.window=pygame.display.set_mode([SCREEN_WIDTH, SCREEN_HEIGHT])
    pygame.display.set_caption('坦克大战 1.01')
    self.createMyTank()
    self.createEnemyTank()
    while True:
        time.sleep(0.02)
        MainGame.window.fill(BG_COLOR)
        self.getEvent()
        #显示文字，把文字写在游戏窗口上
        MainGame.window.blit(self.getTextSurface('游戏说明：'), (820, 10))
        MainGame.window.blit(self.getTextSurface('方向键:控制坦克'), (820, 30))
        MainGame.window.blit(self.getTextSurface('ESC 键：'), (820, 50))
        MainGame.window.blit(self.getTextSurface('复活/重新开始'), (820, 70))
        MainGame.window.blit(self.getTextSurface('Space 键：开炮'), (820, 90))
        if not MainGame.my_tank:
            MainGame.window.blit(self.getTextSurface('您的坦克已被敌方消灭，是否复活？'), (300, 0))
        if len(MainGame.enemyTankList) == 0:
            MainGame.window.blit(self.getTextSurface('游戏胜利，是否重新开始？'), (300, 0))
        self.blitMyTank()
        self.blitEnemyTank()
        self.blitEnemyBullet()
        self.blitMyBullet()
        self.blitExplode()
        if MainGame.my_tank and MainGame.my_tank.live:
            if not MainGame.my_tank.stop:
                MainGame.my_tank.move()
                MainGame.my_tank.myTank_hit_enemyTank()
        pygame.display.update() #将绘制的图像显示在屏幕上
    ..... (省略部分展示过的代码)
    .....

```

综上，坦克大战这个简单的小游戏已经设计完成了。我们介绍坦克大战的游戏的设计的同时也向大家讲解了 Pygame 一些函数的使用方法，希望大家能够掌握如何利用 Pygame 设计小游戏。最终的游戏效果图如下图所示。



图 9-5 游戏效果图

这个游戏有个小缺点就是由于敌方坦克随机移动以及子弹随机发射，这可能会导致游戏开始时我方坦克就被敌方子弹击中。所以读者可以根据自己的设想再完善一下这个小游戏，也可以添加一些音乐等。

9.3 五子棋游戏设计

9.3.1 五子棋简介

我们先来了解一下五子棋。五子棋是全国智力运动会竞技项目之一，是一种两人对弈的纯策略型棋类游戏。通常双方分别使用黑白两色的棋子，下在棋盘直线与横线的交叉点上，先形成 5 子连线者获胜。

1. 棋盘

五子棋专用棋盘为 15×15 ，即为 15 条横线跟 15 条竖线等距排列交叉组成的一个大正方形棋盘。每一条竖线跟横线交叉的点即为下棋的点，总共形成了 225 个下棋点。

2. 棋型和下棋规则

五子棋最常见的基本棋型大体有以下几种：连五，活四，冲四，活三，冲三，活二，冲二。我们将一一介绍这些棋型。

连五：五颗同色棋子连在一起。

活四：有两个连五点（即有两个点可以形成连五），当发现活四出现的时候，如果对方单纯过来防守的话，是已经无法阻止对手连五了。

冲四：有一个连五点的棋型均为冲四棋型；相比于活四来说，冲四的威胁性就小了很多，因为这个时候，对方只要跟着防守在那个唯一的连五点上，冲四就没法形成连五。

活三：可以形成活四的三。活三棋型是我们进攻中最常见的一种，因为活三之后，如果对手不予理会，棋手便可以将活三变成活四，而我们知道活四是已经

无法单纯防守住了。所以，当面对活三的时候，需要非常谨慎对待。在自己没有更好的进攻手段的情况下，需要对其进行防守，以防止其形成可怕的活四棋型。

冲三：只能够形成冲四的三。冲三的棋型与活三的棋型相比，危险系数下降不少，因为冲三棋型即使不去防守，下一手它也只能形成冲四，而对于单纯的冲四棋型，我们知道，是可以防守住的。

9.3.2 人人模式

前一小节介绍了棋盘、棋型和五子棋的基本游戏规则，本节将介绍五子棋游戏的具体实现。本节展示给大家的五子棋游戏主要包含两个界面，游戏主界面和棋盘界面。游戏主界面较为简单，主要是进行模式选择的。主界面提供了两个模式，人人对战模式和人机对战模式。当点击任意一种模式，就会进入下棋界面，也就是棋盘界面。人人对战模式是两个玩家轮流下棋，人机对战模式是一个玩家和电脑轮流下棋。接下来将介绍五子棋游戏中的重点部分的实现。

1. 绘制棋盘界面

五子棋的棋盘界面的构成主要包括两部分，棋盘和按钮。如下图所示。五子棋的棋盘是由纵横各 15 条等距且垂直交叉的平行线构成，在棋盘上有 5 个比较特殊的交叉点，被称为“星”，棋盘正中央的星被称为天元。将棋盘左上角的坐标定为(0, 0)，那么五个星的位置坐标分别为(3, 3), (11, 3), (7, 7), (3, 11), (11, 11)。棋盘界面的按钮提供了开始游戏，放弃游戏以及返回主界面这三种功能。

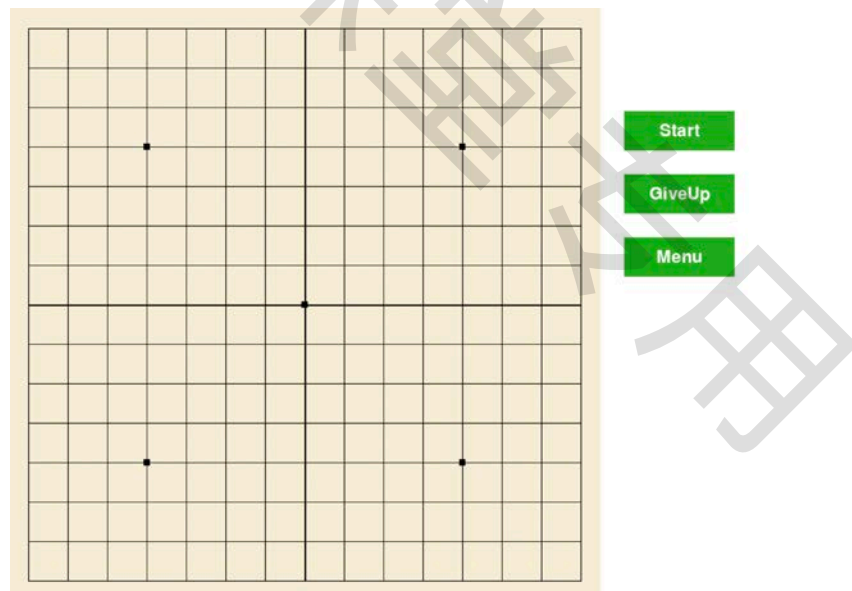


图 9-6 五子棋棋盘界面

```
# 绘制棋盘界面
def draw_background(self):
    color = (0, 0, 0) # 用黑色画棋盘线
    for y in range(self.height):
```



```

# 画水平线, y 坐标相等
    start_pos, end_pos = (REC_SIZE // 2, REC_SIZE // 2 + REC_SIZE * y), \
        (MAP_WIDTH - REC_SIZE // 2, REC_SIZE // 2 + REC_SIZE * y)
    if y == self.height // 2: # 棋盘中间线为粗线
        width = 2
    else:
        width = 1
    pygame.draw.line(self.screen, color, start_pos, end_pos, width)
for x in range(self.width):
    # 画垂直线, x 坐标相等
    start_pos, end_pos = (REC_SIZE // 2 + REC_SIZE * x, REC_SIZE // 2), \
        (REC_SIZE // 2 + REC_SIZE * x, MAP_HEIGHT - REC_SIZE // 2)
    if x == self.width // 2: # 棋盘中间线为粗线
        width = 2
    else:
        width = 1
    pygame.draw.line(self.screen, color, start_pos, end_pos, width)
# 画出棋盘上 5 个特殊点, 即“星”
rec_size = 8
pos = [(3, 3), (11, 3), (3, 11), (11, 11), (7, 7)]
for x, y in pos:
    pygame.draw.rect(self.screen, color,
        (REC_SIZE // 2 + REC_SIZE * x - rec_size // 2,
         REC_SIZE // 2 + REC_SIZE * y - rec_size // 2,
         rec_size, rec_size))
self.draw_button() # 画按钮

```

以上是绘制棋盘界面的代码，棋盘界面的绘制主要包括绘制棋盘，绘制棋盘中的五个“星”以及绘制按钮。其中，全局变量 `REC_SIZE` 表示棋盘中每个方格的大小，`MAP_WIDTH`、`MAP_HEIGHT` 表示棋盘的宽度和高度，具体请参见代码文档。接下来介绍一下在此过程中使用到的 `pygame` 函数。

首先介绍 `pygame` 的 `draw` 模块，该模块包含了一些绘制简单图形的函数，函数的第一个参数是用来指定绘制区域，通常为 `surface` 对象；第二个参数通常是用来表示颜色的 `RGB` 三元组；最后一个参数 `width` 表示绘制线条的宽度，值为 0 时，表示要填充整个图形。下面介绍几个常用的函数：

(1) `pygame.draw.line(surface,color,start_pos,end_pos,width=1)` 该函数用于在 `surface` 上绘制直线段，`start_pos` 和 `end_pos` 以坐标的形式分别表示线段的起点和终点；`width` 默认值为 1。

(2) `pygame.draw.circle(surface,color,pos,radius,width)` 该函数用于绘制圆形，参数 `pos` 是圆心的位置坐标，`radius` 指定了圆的半径。

(3) `pygame.draw.rect(surface,color,rect,width=0)` 该函数用于在 `Surface` 上绘制矩形，参数 `color` 是线条(或填充)的颜色，参数 `rect` 的形式是 `((x,y),(width,height))`，

表示的是所绘制矩形的区域，其中元组(x,y)表示的是该矩形左上角的坐标，元组(width,height)表示的是矩形的宽度和高度；width 表示矩形边的粗细，单位为像素，默认值为 0，表示填充矩形内部。

在我们的游戏界面绘制中，使用 `pygame.draw.line()` 函数来绘制棋盘线，`pygame.draw.rect()` 绘制“星”。绘制完棋盘之后，就开始绘制棋盘界面的按钮部分。按钮的绘制代码参见五子棋游戏代码文档，此处不再列出。

2. 人人对战模式

五子棋人人对战模式进行的方式是两个玩家轮流下棋，玩家每次落子之后都需要根据当前棋局判断是否产生了赢家。如果产生了赢家，游戏结束，打印出赢家信息；如果没有则继续轮流落子。当玩家点击鼠标落子时，需要将鼠标点击位置转换成棋盘上的坐标，并在棋盘界面绘制出相应颜色的五子棋。下面将逐一介绍坐标转换、绘制棋子以及判断输赢的过程。

当玩家点击鼠标进行落子时，首先需要判断鼠标点击坐标的范围，也就是判断鼠标点击坐标是否位于棋盘内，只有当鼠标点击坐标位于棋盘内，才会将鼠标点击坐标转换成棋盘坐标，并记录当前玩家的棋盘坐标，便于以后使用。判断鼠标点击坐标的范围代码如下所示，其中，`map_x`，`map_y` 表示鼠标点击棋盘窗口的坐标，`MAP_WIDTH` 和 `MAP_HEIGHT` 表示棋盘的宽度和高度。

```
# 判断鼠标点击的范围
def is_in_map(map_x, map_y):
    return 0 < map_x < MAP_WIDTH and 0 < map_y < MAP_HEIGHT
```

在 `pygame` 的游戏窗口中的每一个点都有其真实的坐标，而棋盘坐标对应的是一个 15*15 大小的二维坐标系，因此，我们需要获取玩家鼠标点击的真实坐标，然后将其转换为棋盘坐标。定义函数 `MapPosToIndex()` 来完成这个功能，如<示例程序：坐标转换>所示。

```
#<示例程序：坐标转换>
def map_pos_to_index(map_x, map_y):
    return map_x // REC_SIZE, map_y // REC_SIZE
```

全局变量 `REC_SIZE` 表示一个棋格的大小。设置棋盘的边框宽度为 `REC_SIZE/2`。然后设置了一个错位的同样大小的棋盘，如下图所示，错位棋盘的每个格子作为一个判断区域，当鼠标点击该区域时，返回位于当前错位棋盘格中黑色棋盘的顶点坐标。例如当鼠标点击第一行，第一列错位棋盘格时，返回(0, 0)；当鼠标点击第二行，第二列错位棋盘格时，返回(0, 1)。鼠标点击坐标(`map_x, map_y`) 整除以 `REC_SIZE`，即可得到棋盘格上的顶点坐标(x, y)。

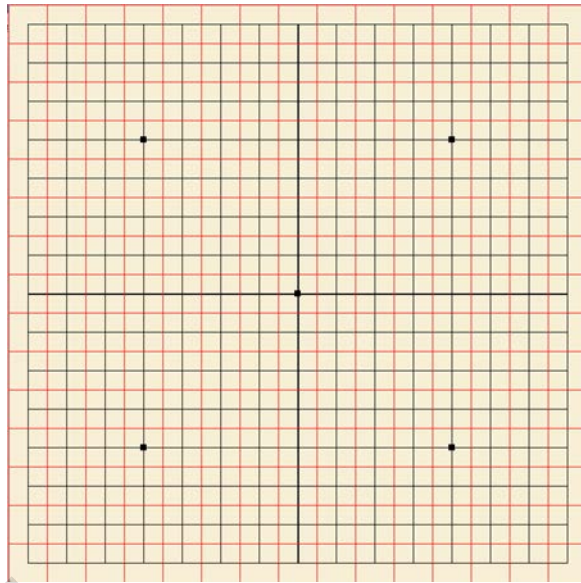


图 9-7 错位棋盘

将玩家鼠标点击坐标转换成棋盘上的坐标之后，在棋盘上画出对应颜色的棋子，即可绘制出当前棋局。我们用 `steps` 列表存储棋盘上已经落子的坐标，用 `map[y][x]` 来标记 `(x, y)` 位置处是黑棋还是白棋。绘制棋子时，遍历 `steps`，将棋盘坐标转化成棋盘的位置坐标，在相应的位置画出棋子即可。绘制棋子的代码如下。

```
#<示例程序：画棋子>
def draw_chess(self, screen):
    player_color = [PLAYER_ONE_COLOR, PLAYER_TWO_COLOR]
    for i in range(len(self.steps)):
        x, y = self.steps[i]
        map_x, map_y, width, height = ChessMap.get_map_unit_rect(x, y)
        pos, radius = (map_x + width // 2, map_y + height // 2), CHESS_RADIUS
        turn = self.map[y][x]
        pygame.draw.circle(screen, player_color[turn - 1], pos, radius)
```

接下来将介绍如何判断输赢。在每次落子结束后，需要对当前棋局的输赢情况进行判断，有一个简单粗暴的方法是，遍历当前棋盘上的每一个落子点，以该点为中心沿着横向、纵向、两个对角线方向搜索是否有活五的情况，也就是判断是否有五个相同颜色的棋子连成一线。当每次玩家落子之后，都对当前棋盘上的所有落子点进行一次遍历来查找活五没有必要，而且效率也很低。我们注意到，无论是玩家落子还是电脑落子，其目的必然是围绕当前落子点构建一个五子相连的胜利条件，所以判断胜负只需要围绕玩家当前落子点进行即可，而不需要遍历当前棋局的所有落子点。以当前落子点为中心进行横向、纵向、对角线方向搜索时，如果出现活五就终止对战，结束游戏。`is_win()`函数实现了判断输赢的功能，`WhetherExistLiveFive()`函数来判断是否存在连五。代码实现如<示例程序：游戏

结果的判定>所示。

#<示例程序：游戏结果的判定>

```
def is_win(self, board, mine, x, y):
    dir_offset = [(1, 0), (0, 1), (1, 1), (1, -1)] # direction
    for i in range(4):
        result = self.WhetherExistLiveFive(board, x, y, dir_offset[i], mine)
        if result:
            return True
    return False
```

判断是否有连五的情况

```
def WhetherExistLiveFive(self, board, x, y, dir, mine):
    # 获取 dir 方向上的落子情况
    line = self.getLine(board, x, y, dir)
    # 对(x, y)点的 dir 方向的 5 个五元组进行遍历
    flag = True
    for i in range(5):
        left_idx, right_idx = 3, 5

        while left_idx >= i:
            if line[left_idx] != mine:
                flag = False
                break
            left_idx -= 1

        while right_idx <= i + 4 and flag == True:
            if line[right_idx] != mine:
                flag = False
                break
            right_idx += 1

    if flag:
        return True
    else:
        flag = True
    return False
```

在<示例程序：游戏结果的判定>中，`is_win()`函数中的 `dir_offset` 表示要查找的每个落子点的四个方向，横向、纵向、两个对角线方向。判断输赢时，以当前落子点为中心分别对横向、纵向、对角线方向这四个方向搜索，看是否有连五的情况。对于是否有连五的情况的判断，首先调用 `getLine()`函数获得当前方向 `dir` 上以 `(x,y)` 为中心的 9 个位置的落子情况，存放于 `line` 列表中，其中 `line` 的大小为 9。然后根据 `line` 列表，对 `(x, y)` 点的 `dir` 方向的 5 个五元组进行遍历，判断每个五元组中黑子和白子的数量情况，当存在一个五元组都是黑子或者白子的时候，就产生了赢家。对于五元组和 `line` 列表的解释如下图所示。

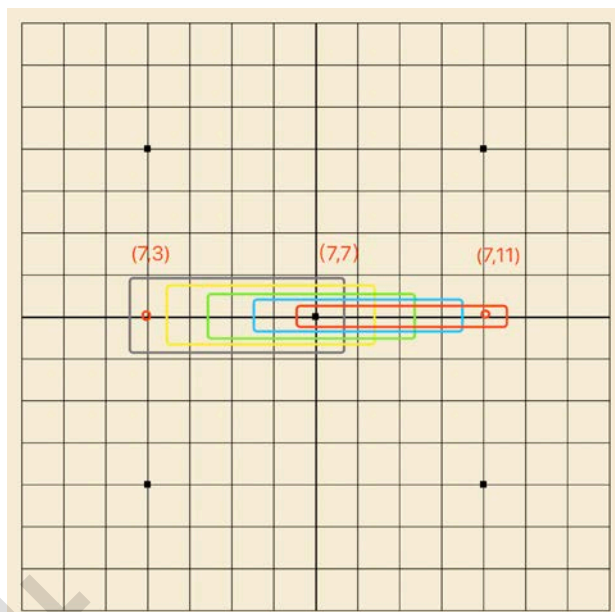


图 9-8 五元组

如图 9-8 中所示，假设当前玩家落子点为 (7,7)，中间的黑点表示 (7,7)，两边的红色点表示 (7,3) 和 (7,11)。当判断横向这个方向是否有连五时，首先先把以 (7,7) 为中心的 9 个点的落子情况存到列表 `line` 中。也就是会把横向方向上从 (7,3) 到 (7,11) 这一条线段上的 9 个点的落子情况存到列表 `line` 中。例如，当 (7,3) 位置是个黑子，则 `line[0]=1`；当 (7,4) 位置对应的是白子，则 `line[1]=2`；当某个位置没有落子时，对应的 `line` 就是 0。我们将棋盘每五个连续的位置看作是 1 个五元组，以 (7,7) 为中心的 9 个点组成了 5 个五元组，分别是 [(7,3), (7,4), (7,5), (7,6), (7,7)]、[(7,4), (7,5), (7,6), (7,7), (7,8)]、[(7,5), (7,6), (7,7), (7,8), (7,9)]、[(7,6), (7,7), (7,8), (7,9), (7,10)]、[(7,7), (7,8), (7,9), (7,10), (7,11)]。每一个五元组中黑子和白字的数量可能都不一样，判断输赢时，就分别判断这 5 个五元组中是否有颜色相同的五连子即可。这个例子横向搜索时的情况，其他三个方向和此类似。

9.3.3 人机模式

接下来介绍一下五子棋游戏的人机模式。人机模式就是玩家和电脑轮流下棋，每次落子之后，都要根据当前落子位置来判断输赢。人机模式和人人模式判断输赢的方法是一样，都是使用 `is_win()` 函数，上一小节已经介绍过。那么，电脑下棋时是如何确定它的落子位置呢？接下来我们将介绍最重要的部分-电脑落子决策，也就是电脑是如何落子的。

电脑落子决策的基本思想是对棋局的有效位置进行评分，根据某个有效位置的得分来决定落子位置。本节介绍普通难度下的落子决策，即电脑只考虑当前局面下的最优解来落子，我们称之为评分表策略。电脑落子的基本步骤是，首先先获得棋盘上的所有空位坐标，也就是棋盘上的所有未落子点；然后对每一个空位

进行打分，找到所有空位中分数最大的空位，记录这个最大分数和空位的坐标，即是电脑的落子位置。代码实现如<示例程序：电脑落子>所示。

```
#<示例程序：电脑落子>
def find_best_chess(self, board, turn):
    moves = self.getEmpty(board) # 获得棋盘中未落子坐标
    bestmove = None
    max_score = -0x7fffffff
    if turn == MapEntryType.MAP_PLAYER_ONE:
        mine = 1
        opponent = 2
    else:
        mine = 2
        opponent = 1
    for score, x, y in moves:
        # 对每一个空位进行打分
        score = self.evaluatePoint(board, x, y, mine, opponent)
        if score > max_score:
            max_score = score
            bestmove = (x, y) # 记录取得最大分数的坐标
    return bestmove
```

那么，如何计算每个空位的分数呢？每个空位的得分是这个空位的横向、纵向、两个对角线方向的这四个方向所包含的五元组的得分之和。以某个空位点为中心，每个方向上都包含了 5 个五元组。针对五元组中黑子和白子的数量的不同，评分表会给出不同的分数。所以，需要计算出四个方向上的得分，累加就是这个空位的得分。

计算空位点的分数的计算过程如图 9-9 计算空位点分数示意图所示，对于空位点 1，先统计以它为一端的矩形圈起来的五元组，即图中最小的矩形，然后矩形依次移位，直到空位点 1 处于矩形的另一端，这样我们就计算出了空位点 1 横向的分数，接着计算纵向和斜向的分数，相加即得到了空位 1 当前棋局下的得分。

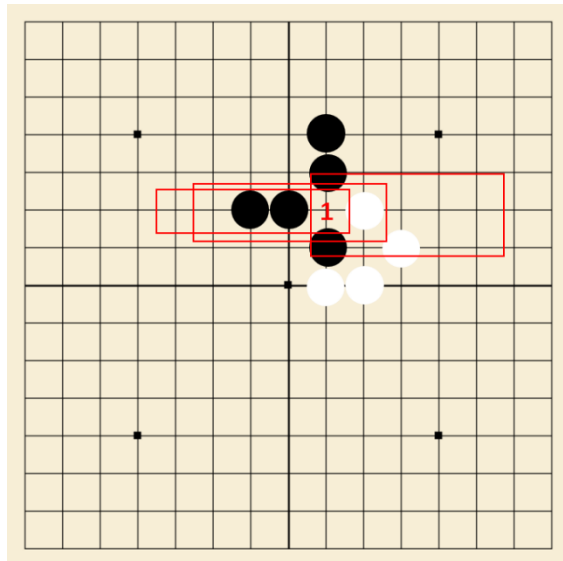


图 9-9 计算空位点分数示意图

计算空位得分的代码实现如<示例程序：计算空位的得分>所示。

#<示例程序：计算空位的得分>

```
def evaluatePoint(self, board, x, y, mine, opponent):
    dir_offset = [(1, 0), (0, 1), (1, 1), (1, -1)] # direction
    score = 0
    for i in range(4):
        # 四个方向上的得分累加
        score += self.analysisLine(board, x, y, dir_offset[i], mine, opponent)
    return score
```

计算每个方向上的得分

```
def analysisLine(self, board, x, y, dir, mine, opponent):
    line = self.getLine(board, x, y, dir)
    score_count = 0 # 存放 dir 方向上 5 个五元组的评分总和
```

```
# 对(x, y)点的 dir 方向的 5 个五元组进行遍历
# mine_chess_sum 为该五元组中白色棋子的数目
# opponent_chess_sum 为该五元组中黑色棋子的数目
```

```
for i in range(5):
    left_idx, right_idx = 3, 5
    mine_chess_num = 0
    opponent_chess_sum = 0

    while left_idx >= i:
        if line[left_idx] == MapEntryType.MAP_NONE:
            break;
        if line[left_idx] == mine:
            mine_chess_num += 1
        if line[left_idx] == opponent:
            opponent_chess_sum += 1
        left_idx -= 1
```

```

while right_idx <= i + 4:
    if line[right_idx] == MapEntryType.MAP_NONE:
        break;
    if line[right_idx] == mine:
        mine_chess_num += 1
    if line[right_idx] == opponent:
        opponent_chess_sum += 1
    right_idx += 1

# 根据评分表进行评分
score_count += self.getScore(mine_chess_num, opponent_chess_sum)

return score_count

```

在#<示例程序：计算空位的得分> 中，analysisLine()函数是用来计算每个方向上的五元组的得分。分别遍历每个五元组，用变量 mine_chess_sum 和 opponent_chess_num 来记录每个五元组中包含的黑子和白子的数量，然后根据 mine_chess_sum 和 opponent_chess_num 的关系查找评分表，对五元组评分。最后将每个五元组的得分累加即得到了某个方向上的得分。

如何根据每个五元组的黑白棋数量打分呢？这个分数就是根据我们制定的评分表来打分的。评分表的分数是可以自己设定，分数需要区分不同局势的差异性，不同的评分机制可能带来不同的效果，<示例程序：评分表>展示了本节预设的评分表。

```

#<示例程序：评分表>
class SCORE2(IntEnum):
    SCORE_NONE = 7,
    SCORE_ONE_BLACK = 35,          #只有一个黑子
    SCORE_TWO_BLACK = 800,
    SCORE_THREE_BLACK = 15000,
    SCORE_FOUR_BLACK = 800000,
    SCORE_ONE_WHITE = 15,         #只有一个白子
    SCORE_TWO_WHITE = 400,
    SCORE_THREE_WHITE = 8000,
    SCORE_FOUR_WHITE = 100000,
    SCORE_BOTH = 0,              # 黑白子混合
    SCORE_MAX = 0x7fffffff,
    SCORE_MIN = -1 * 0x7fffffff,

```

评分表中，SCORE_NONE 对应没有棋子时的得分，之所以没有设定为 0，是因为还有比没有棋子更糟糕的局势，例如五元组中有黑白子混合的情况。其他的表示五元组只含有若干个黑子或者白子的得分，SCORE_BOTH 代表黑白子共存时的得分，五元组中有黑白子混合时得分为 0。所以，根据评分表就可以对五元组中的不用的黑白子数量进行评分。评分代码如<示例程序：五元组评分>所示。


```
#<示例程序：评分表>
def getScore(self, white_chess_count, black_chess_count):

    # 没有子
    if white_chess_count == 0 and black_chess_count == 0:
        return SCORE2.SCORE_NONE
    # 黑白子共存
    if white_chess_count != 0 and black_chess_count != 0:
        return SCORE2.SCORE_BOTH
    # 仅仅含有黑子
    if black_chess_count == 1 and white_chess_count == 0:
        return SCORE2.SCORE_ONE_BLACK
    if black_chess_count == 2 and white_chess_count == 0:
        return SCORE2.SCORE_TWO_BLACK
    if black_chess_count == 3 and white_chess_count == 0:
        return SCORE2.SCORE_THREE_BLACK
    if black_chess_count == 4 and white_chess_count == 0:
        return SCORE2.SCORE_FOUR_BLACK
    # 仅仅含有白子
    if white_chess_count == 1 and black_chess_count == 0:
        return SCORE2.SCORE_ONE_WHITE
    if white_chess_count == 2 and black_chess_count == 0:
        return SCORE2.SCORE_TWO_WHITE
    if white_chess_count == 3 and black_chess_count == 0:
        return SCORE2.SCORE_THREE_WHITE
    if white_chess_count == 4 and black_chess_count == 0:
        return SCORE2.SCORE_FOUR_WHITE
```

至此，普通模式下的五子棋落子决策介绍完毕，尽管评分表策略简单且表现不错，但是它只考虑了当前棋局中最好的情况，只进行了一层搜索。在现实中，每一步的选择很有可能会影响到未来几步的棋局情况，需要更深层次的搜索，下一节我们将介绍 Alpha-Beta 剪枝搜索算法，学习利用博弈的思想来解决落子决策问题。

9.3.4 Alpha-Beta 剪枝搜索算法

下棋是博弈的一种，博弈的过程可以用一棵博弈搜索树表示，树中每一个节点代表棋盘上的一个局面，通过对搜索树进行搜索可以求取问题的解，本节搜索策略采用 Alpha-Beta 剪枝搜索，这是一个常用于人机游戏对抗的搜索算法。接下来，我们通过探究 Alpha-Beta 剪枝搜索算法来看计算机是如何下棋的。

在下棋时，博弈树的每个子节点代表一个可能的落子点，对于每个子节点，通过计算其效用值来代表某一方选择该落子点的有利分数值。考虑 A 与 B 两方下棋，A 在落子时，要选择对自己最有利的落子点，即在落子时对其子节点的效

用值总是取极大值，将 B 定为 MIN 方，由于效用值代表对 A 最为有利的分数，因此，B 在落子时总是对其子节点的效用值取极小值，即选择对 A 最不利的落子点。模拟博弈的过程，MAX 和 MIN 交互构建出一棵博弈搜索树。

在本节博弈搜索树的图例中，我们用“正三角形”代表 MAX 节点，表示轮到 MAX 方下棋；用“倒三角形”代表 MIN 节点，表示轮到 MIN 方下棋。终端节点值是对 MAX 方的效用值，其他节点用它们子节点的最大或者最小值标记。效用值总是从 MAX 方的角度来看的，高值对 MAX 方有好处，对 MIN 方有坏处。当 MAX 方走棋时，试图使效用值最大，即最大化自己的表现；当 MIN 方走棋时，使效用值最小，试图贬低 MAX 方的表现。在搜寻时，MAX 节点所记录的当前最大有效值称为 Alpha 值，MIN 节点所记录当前的最小值称为 Beta 值。Alpha 值可能会随着搜寻而改变，当子节点有新的值比当前的 Alpha 值大时，Alpha 值会取较大值。明显的，Alpha 值只会随着搜寻变大或不变，而 Beta 值只可能变小或不变。

Alpha-Beta 剪枝算法是一个基于剪枝的深度优先搜索算法，旨在减少上述极大极小值搜索算法中需要做评估的节点数。它用于裁剪搜索树中不需要搜索的树枝，以提高搜索速度。它的基本思想是根据上一层已经得到的当前最优结果，决定目前的搜索是否要继续下去。Alpha-Beta 剪枝算法的基本原理是：

- (1) 当一个 MIN 节点的 β 值小于等于其任何一个父节点 (MAX 节点) 的 α 值时，减掉该节点的所有子节点。
- (2) 当一个 MAX 节点的 α 值大于等于任何一个父节点 (MIN 节点) 的 β 值时，减掉该节点的所有子节点。

剪枝算法的原理是很简单的，对 MAX 节点而言，假如其某一个子节点的效用值 Beta 比它父节点的 Alpha 值小，由于 Beta 值只可能会变小，它将来没有可能大于父节点的 Alpha 值了。我们就不需要持续搜索此子节点下面的任何分支了，称为“裁剪”此分枝。同理，对 MIN 节点而言，也可以根据子节点的 Alpha 值对自己的 Beta 值来比较作可能裁剪。这二类裁剪会大量减少搜寻量。

在对博弈树采取深度优先的搜索策略时，从左路分枝的叶节点倒推得到某一层 MAX 节点的值，可表示目前落子点的最佳效用值，记为 α 。此值可作为 MAX 方落子效用值的下界。在搜索其它子节点时，如果发现一个回合 (2 步棋) 之后效用值变差，即子节点效用值低于下界 α 值，则可以剪掉此枝 (以该子节点为根的子树)，不再考虑往此子节点的延伸，此类剪枝称为 α 剪枝，如图 9-10 α 剪枝示意图所示。

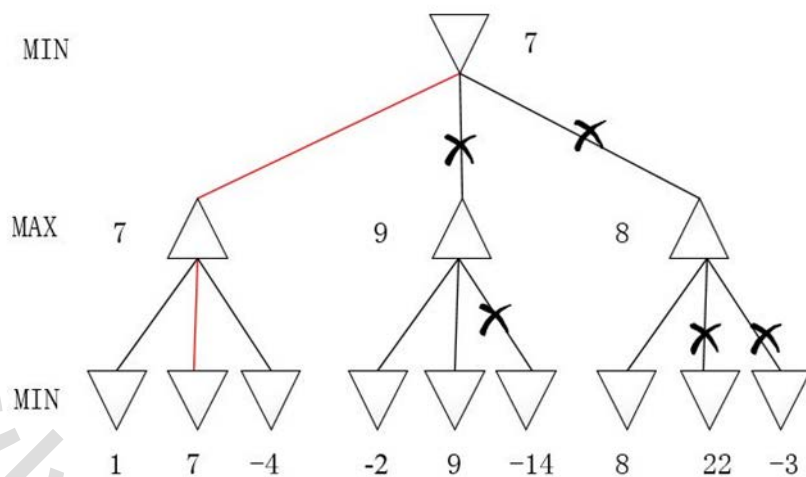


图 9-10 α 剪枝示意图

由左路分枝的叶节点倒推得到某一层 MIN 节点的值，可表示目前对对方落子的钳制值，记为 β 。 β 值可作为 MAX 方无法实现落子指标的上界。在搜索该 MIN 节点的其它子节点时，如果发现一个回合之后钳制局面减弱，即子节点效用值高于上界 β 值，则可以剪掉此枝，即不再考虑此子节点的延伸。此类剪枝称为 β 剪枝，如图 9-11 β 剪枝所示。

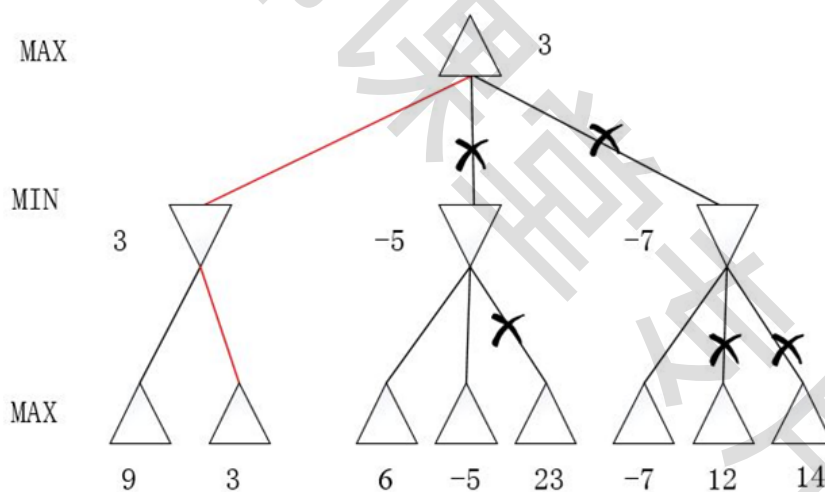


图 9-11 β 剪枝示意图

Alpha-Beta 剪枝搜索算法介绍完毕，在该算法中，计算节点效用值的函数称为评估函数，节点的分数的不固定的，不同的分数代表着不同的棋局局势，9.3.1 节中介绍了几种不同的棋型：活三、冲三、活四、冲四等，在评分时，有“越靠近 5 越好，越活越好”的规律，请同学们思考，Alpha-Beta 剪枝搜索算法中的评估函数应该如何设计。

习题

习题 9.1 利用 Pygame 绘制移动的矩形。利用 `pygame.draw.rect` 函数绘制出一个矩形，并使该矩形在游戏窗口范围内随机移动。当该矩形遇到窗口边界时，使其弹回继续随机移动。

习题 9.2 利用 Pygame 的 `draw` 模块绘制如图 9-12 所示的小丑脸图形。其中，眼睛是两个实心圆形，填充为蓝色；鼻子是椭圆形，填充为粉红色；嘴巴处是弧形，填充为红色。



图 9-12 小丑脸

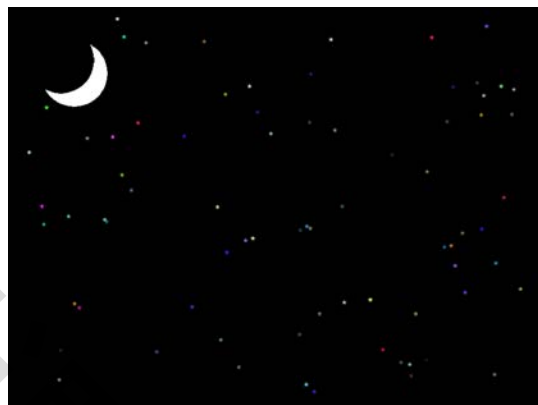


图 9-13 满天星

习题 9.3 利用 Pygame 的 `mouse` 模块实现下面的游戏效果：（1）当鼠标经过窗口时，窗口背景颜色会随着鼠标的移动而发生改变（2）当鼠标左键点击窗口时，在控制台打印出“The left button is pressed !”；当鼠标右键点击窗口时，在控制台打印出“The right button is pressed !”；当鼠标滚轮被按下时，在控制台打印出“The wheel button is pressed !”。

习题 9.4 绘制小游戏满天星。随机产生 100 颗小星星，小星星从窗口的左上角到右下角循环移动，使得小星星的颜色随机并闪烁，同时窗口左上角要绘制出月亮。效果图如图 9-13 所示。

习题 9.5 使用 Pygame 制作一个不断移动的小球。在游戏窗口中创建一个不断斜向移动的小球，并在小球移动至窗口边缘时使其弹回并继续移动。

习题 9.6 在习题 9.5 的基础上添加鼠标和键盘控制事件，使用键盘加快小球移动的速度；使用鼠标控制小球的位置。

习题 9.7 利用 Pygame 绘制烟花。

习题 9.8 9.2 节中详细介绍了坦克大战游戏的设计过程，请在该游戏的基础上添加背景音乐。

习题 9.9 利用 Pygame 开发一个贪吃蛇小游戏，通过键盘的方向键“上下左右”来控制贪吃蛇的移动。

习题 9.10 请使用 Pygame 实现一个任意类型的迷宫类游戏。

习题 9.11 2048 小游戏简单又有趣，玩家选择上下左右四个方向其中一个滑动数字块，每次滑动会使得所有数字块朝着滑动的方向靠拢，系统在空白的地方生成一个值为 2 或 4 的数字方块，相同数字的方块在靠拢、相撞时会相加。玩家需要想办法在 16 格范围中凑出值为“2048”的数字块，若 16 格填满还未凑出 2048 则判定游戏失败。游戏界面如图所示，请使用 Pygame 实现 2048 小游戏。



9-14 2048 小游戏

习题 9.12 在 9.3.3 节介绍了计算空位分数的评分表策略，空位分数需要体现出当前局势的好坏，请设计自己的评分表和空位分数计算方式，实现五子棋游戏。

习题 9.13 在 9.3.4 节介绍了 Alpha-Beta 剪枝搜索算法，其中计算节点效用值的评估函数的设计是非常重要的，它需要体现不同的局势，请利用 Alpha-Beta 剪枝搜索算法设计一种评估函数。

习题 9.14 请应用习题 14 中你设计的评估函数，实现五子棋游戏的人机对战模式。

习题 9.15 请从游戏开发者的角度谈谈，一个有趣的游戏需要具备哪些特征，在设计游戏时需要注意什么，请发挥你的想象力，试着设计一个新颖且有趣的游戏吧。