

换个思路挑战“小恐龙”游戏

谢作如 浙江省温州中学
林淼焱 浙江省温州外国语学校

很多人都知道,Chrome浏览器在出现无网络连接时会提供一个恐龙小游戏,帮我们缓解一下情绪。只要按下空格键,这只小恐龙就会动起来,开始在沙漠中狂奔,然后会有仙人掌、飞鸟等源源不断地迎面而来,玩家需要操控小恐龙跳跃到空中,以躲避这些障碍物(如图1)。在浏览器地址栏输入chrome://dino/,联网状态下也可以玩这个游戏。

因为受网络控制,学生在机房会经常看到这个页面,也会经常玩这个游戏。这给我们一个灵感:为什么不让学生试试用Python代码编写一个自动玩游戏的脚本,然后比赛谁写的代码效果好呢?这可是一个有挑战性的任务。

● 解决方案分析

能不能用Python脚本自动“玩”这个浏览器游戏?答案不言而喻,借助Python强大的第三方库,不仅可以对障碍物进行识别,还能模拟键盘输入,让小恐龙跳起来,达到我们想要的自动玩游戏的效果。

用流程图的形式来描述玩小恐龙游戏的过程,只有两个核心的工作流程,即判断前方是否有障碍和按下空格躲避障碍,用流程图来描述,如图下页2所示。

接下来,分析这个游戏的玩法,要想实现自动“玩”,我们需要解决三个问题:①如何获得当前的障碍物信息?②如何模拟按键让小恐龙跳起来?③如何在适当的时间跳起来?

问题②是最容易解决的。Python有很多库可以模拟键盘,如在Windows系统下,常用的是调用win32api库,非Windows系

统的计算机可以用pykeyboard。但是考虑到判断障碍物需要用到截图,而截图方面比较好用的库是pyautogui,因此选择用pyautogui的typewrite方法。代码如下:

```
pyautogui.typewrite(message=' ')
```

引号中的字符就是要模拟的按键,如果是空格,那么就在引号中放一个空格。也可以用“pyautogui.press("space")”的形式,“space”表示空格键。

所以,这个工作流程中的核心算法,是问题①和③,即判断障碍物信息,并在合适时间跳起来。



图1

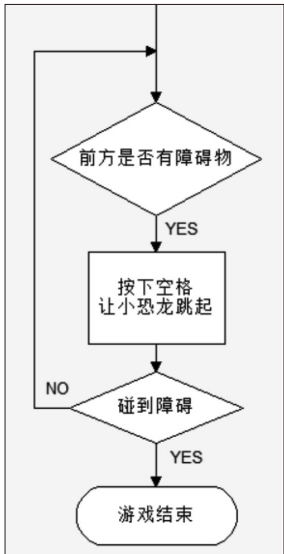


图2

● 核心算法的解决

观察游戏界面可以知道,如果小恐龙“眼前”的区域都是白色,那么说明没有障碍物,如果出现了其他颜色,说明障碍物来了,如图3中框选区域所示。要想让计算机自动判断是否有障碍物,可以用定时截图的方式,然后不断“识别”框中区域的颜色,再根据识别结果来确定是否按下空格键。具体实现方法如下。

1.截图并且识别障碍物

通过pyautogui库完成屏幕截图后,再借助PIL库,来识别目标区域的颜色。为了加快判断的速度,我们可以继续裁剪图像,裁去场景中的云朵、地面等干扰物,只剩下框中的内容。

识别图片的颜色,大家想到的方法可能是读取图片中每个像素点的数据,将其相加并求出平均数,如果这个数字小于255,说明图片不是纯白色,包含黑色的障碍物。其实还有一种更简单的方法,即利用PIL库

中的getcolors方法,统计整张图片中所有像素点的颜色及其数量。

对于Python来说,要想提高执行代码的效率,应该尽可能调用库中的方法。getcolors方法会返回一个列表,如[[(颜色1像素点的数量), (颜色1RGB值)]]。图4中的案例结果就代表图片中共有68751个白色像素点、3011个绿色像素点、1958个红色像素点、2522个蓝色像素点。

有了这个方法,判断颜色就非常简单了。用getcolors方法返回颜色的列表,再用len()函数就能得到图片中所有颜色的种类数。没有障碍物时,颜色是1,即只有白色,大于1就说明有障碍物了。参考代码如下页图5所示。

2.选择适当的跳跃时机

当小恐龙“眼前”不止一种颜

色时,说明障碍物已经接近,需要让小恐龙及时跳跃,进行躲避。

一般来说,用下页图6这样的语句就能让小恐龙跳过很多障碍物,并且得到很高的分数了。但是,当速度越来越快的时候,是不是还要继续判断列表中的白色数量?因为白色的数量越小,说明障碍物已经完全进入了区域。或者是否要多划分几个区域,然后得到障碍物离小恐龙的位置?这就要不断调试,找到最适合的时机。

● 代码实现和测试

将上述代码合并在一起,添加循环,使得程序能够不断进行判断,这个能够自动控制小恐龙的“外挂”代码就写好了。参考代码如下页图7所示。

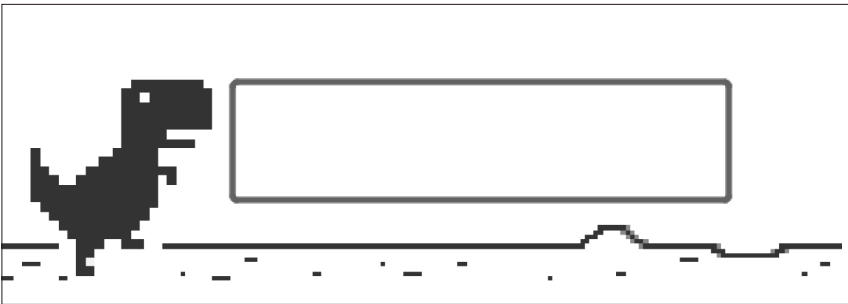


图3

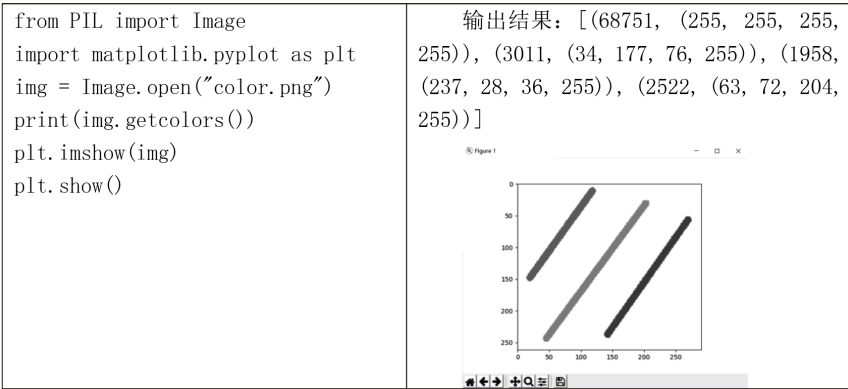


图4

```
import pyautogui
from PIL import Image
img=pyautogui.screenshot()
#得到图片的宽高
height,width=img.height,img.width
#裁剪图片(left, upper, right, lower), 以图片的左上角和右下角作为标准
cropped = img.crop((int(width*0.12),int(height*0.56), int(width*0.3),
int(height*0.65)))
colors=cropped.getcolors()
print("颜色的种类为",len(colors))
```

输出结果：颜色的种类为 1

图5

```
import pyautogui
from PIL import Image
while True:
    img=pyautogui.screenshot()
    height,width=img.height,img.width
    box=(int(width*0.12),int(height*0.56), int(width*0.3), int(height*0.65))
    cropped=img.crop(box)
    colors=cropped.getcolors()
    print(len(colors))
    if len(colors)!=1:
        pyautogui.press("space")
```




图7

```
draw = ImageDraw.Draw(img)
box=(int(width*0.12),int(height*0.56), int(width*0.3), int(height*0.65))
draw.rectangle(box, fill=None, outline='red',width=5)
img.show()
```

图8

因为不同的计算机显示屏的大小也是不一样的,我们要根据分辨率微调截图参数,以达到最好的效果。为了更好地确定位置,可以用PIL库的ImageDraw模块来画出区域,代码如图8所示。

经过测试,我们这段简单的代码居然可以拿到高于1000的分数。

这是我们平时根本没办法达到的分数。当看到小恐龙在屏幕上不断自动跳跃,一种特别的满足感会油然而生。

● 算法的优化

当达到一定分数后,这个小游戏的场景会在黑夜与白天之间切换,且障碍物移动速度增加,大大

```
if len(colors)!=1:
    pyautogui.press("space")
```

图6

增加了游戏的难度。上述代码最高能够到达1500分左右,由于障碍物的提速,会让小恐龙在落地前就撞上下一个障碍物,想要继续提高分数,就需要继续优化代码。比如,①让小恐龙看得更远:随着分数的增加,不断扩大截取图片的范围,直到这个范围增加到屏幕右端为止。②提前预判起跳时间:在图片的中间和右端各截取一张图片,根据障碍物的通过时间,计算其速度,列出算式,求得障碍物到达小恐龙前方的时间,小恐龙在等待相应的延时后,精准起跳。Pyautogui库中还有keyDown和keyUp的控制键盘方法,可以实现短按和长按的效果。

● 总结

《普通高中信息技术课程标准(2017年版)》在必修模块1中提到,要让学生“通过解决实际问题,体验程序设计的基本流程,感受算法效率,掌握程序调试与运行的方法”,而控制游戏这种有趣的案例,既源于生活又具有不断优化特点,寓教于乐,很适合作为教学的拓展案例,让学生在探索过程中提高编程能力。此外,这种“自动化”的思想,能够引发学生对人工智能的学习兴趣,值得在教学中推广。e