Program:Lifegame

目录：一．Profile—性能分析工具

二．Profile简介

三．Profile在我们程序中的具体应用

四．测试结果以及程序优化

1. Profile 性能分析工具

程序运行慢的原因有很多种，比如存在大量的劣化代码，但真正的原因往往是一两段设计的并不那么好的程序，比如一序列元素进行自定义类型的转换等，在程序中我们很难靠经验找出造成性能瓶颈的代码，而对代码优化的前提是我们要了解性能的瓶颈在什么地方，这个时候我们就要借助一个好用的性能分析工具--profile。

1. Profile简介：

Profile被包含在python的标准库中，可以统计程序中每个函数的运行时间,profile的统计分析分为以下几个方面：

ncalls：表示函数调用的次数；

tottime：表示指定函数的总的运行时间，除掉函数中调用子函数的运行时间；

percall：（第一个 percall）等于 tottime/ncalls；

cumtime：表示该函数及其所有子函数的调用运行的时间，即函数开始调用到返回的时间；

percall：（第二个 percall）即函数运行一次的平均时间，等于 cumtime/ncalls；

filename:lineno(function)：每个函数调用的具体信息；

如果需要将输出以日志的形式保存，只需要在调用的时候加入另外一个参数。如 profile.run(“profileTest()”,”testprof”)。

用pstats自定义报表

profile解决了我们的一个需求，还有一个需求：以多种形式查看输出，我们可以通过profile的另一个类Stats来解决。我们可以引入一个模块pstats，它定义了一个类Stats，Stats的构造函数接受一个参数——就是profile的输出文件的文件名。Stats提供了对profile输出结果进行排序、输出控制等功能。

三．使用Profile对我们的程序进行分析

首先将我们的程序分成几个相对独立的部分，方便进行我们的性能测试

1. Button的部分：里面包含一个Button类，类里面具体包含init初始化的方法以及drawbutton方法，用来指定按钮和图像位置。
2. Cell的部分：定义Cell类，主要用来初始化细胞，设定细胞的大小和颜色，获取细胞位置在相应位置绘制细胞
3. Data的部分：用来存储和设置画布的宽和高的具体数值
4. Draw的部分：包含draw的方法，将画布擦除后，用来绘制整个地图的网格线，以及遍历网格组数组中的细胞状态
5. Main的部分：整个程序的开始入口，定义状态机的三种状态，分别是初始化，停止和进行，然后在整个函数里面执行游戏的主循环

f.Next\_generation:具体的包含next\_generation方法，定义细胞更新的规则：当一个细胞周围有两个或三个细胞时细胞状态为存活，当周围细胞过多或者过少时定义为细胞死亡

g.States的部分：将整个游戏世界设定为二维数组，每个网格状态为0和1两种

包含init（初始化）,move（开始运行）和stop（暂停运行）方法，分别为：

对整个画布进行初始化，用0填充数组；

定义暂停时的操作，根据鼠标的状态随时更细画布；

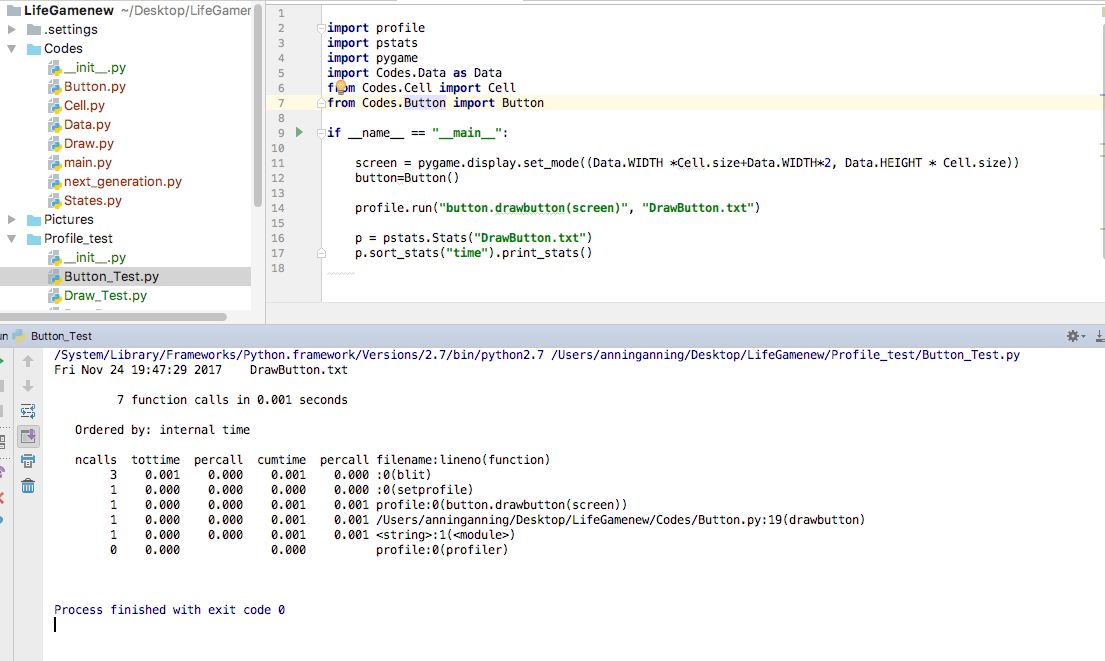
以及进行演化时的操作，获取鼠标当前状态和位置并随时更新画布，再根据细胞更新规则随时对细胞进行更新

说明:我们根据profile测试程序性能时的特点，我们依次根据我们完成整个程序的每个方法进行测试，而Cell部分和Main部分则无需进行性能测试，因为他们已经是最优化的状态，我们没有可优化的空间，进行多余的测试是没有必要的，而对于States我们将分为两个部分对其两个方法进行测试，以对其进行更好的测试以及后续的优化

四．测试代码与优化结果

Button\_Test测试代码：

**import** profile  
**import** pstats  
**import** pygame  
**import** Codes.Data **as** Data  
**from** Codes.Cell **import** Cell  
**from** Codes.Button **import** Button  
   
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
   
 screen = pygame.display.set\_mode((Data.WIDTH \*Cell.size+Data.WIDTH\*2, Data.HEIGHT \* Cell.size))  
 button=Button()  
   
 profile.run(**"button.drawbutton(screen)"**, **"DrawButton.txt"**)  
   
 p = pstats.Stats(**"DrawButton.txt"**)  
 p.sort\_stats(**"time"**).print\_stats()

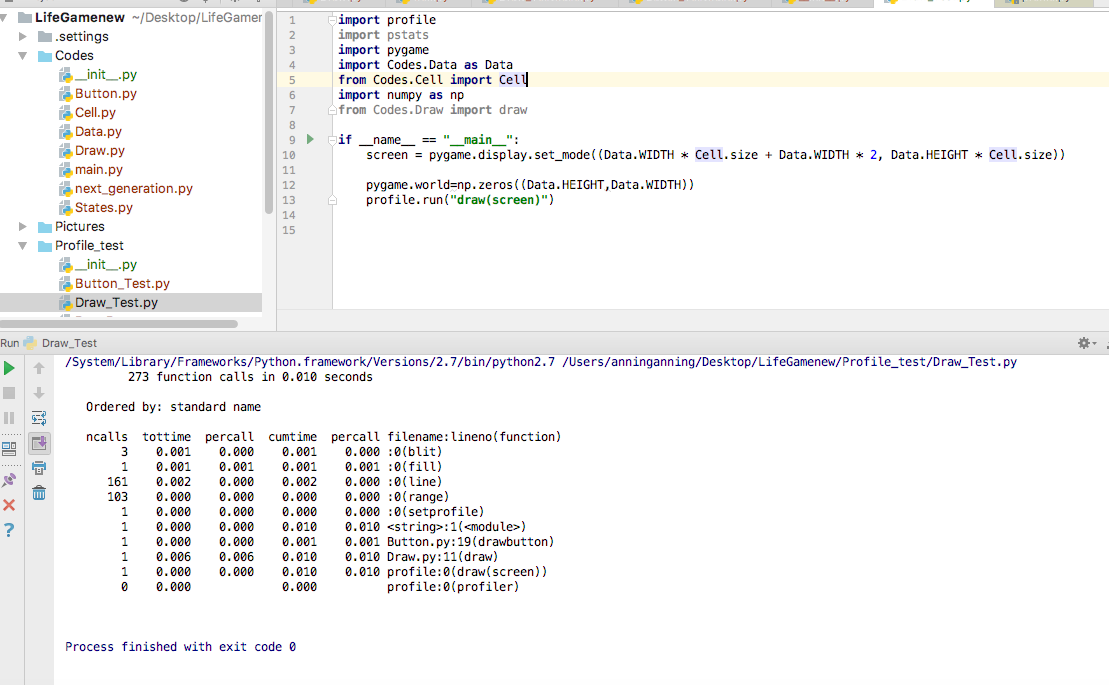


Button.py 源代码

*#-\*- coding:utf-8 –\*-***import** pygame  
**import** Data  
**from** Cell **import** Cell  
  
*#按钮类***class** Button():  
 **def** \_\_init\_\_(self): *#指定按钮的图像和位置* self.imageMove= pygame.image.load(**'../Pictures/Move.PNG'**)*#进行演化* self.imageStop= pygame.image.load(**'../Pictures/Stop.PNG'**)*#停止* self.imageReset= pygame.image.load(**'../Pictures/Reset.PNG'**)*#重置地图  
   
 #设置图片位置* self.positionMove = ((2+Data.WIDTH)\*Cell.size,Data.HEIGHT)  
 self.positionStop = ((2+Data.WIDTH)\*Cell.size,Data.HEIGHT\*3)  
 self.positionReset = ((2+Data.WIDTH)\*Cell.size,Data.HEIGHT\*5)  
   
 **def** drawbutton(self,screen): *#画按钮* screen.blit(self.imageMove,self.positionMove)  
 screen.blit(self.imageStop,self.positionStop)  
 screen.blit(self.imageReset,self.positionReset)

Draw\_Test测试源代码：

**import** profile  
**import** pstats  
**import** pygame  
**import** Codes.Data **as** Data  
**from** Codes.Cell **import** Cell  
**import** numpy **as** np  
**from** Codes.Draw **import** draw  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 screen = pygame.display.set\_mode((Data.WIDTH \* Cell.size + Data.WIDTH \* 2, Data.HEIGHT \* Cell.size))  
  
 pygame.world=np.zeros((Data.HEIGHT,Data.WIDTH))  
 profile.run(**"draw(screen)"**)

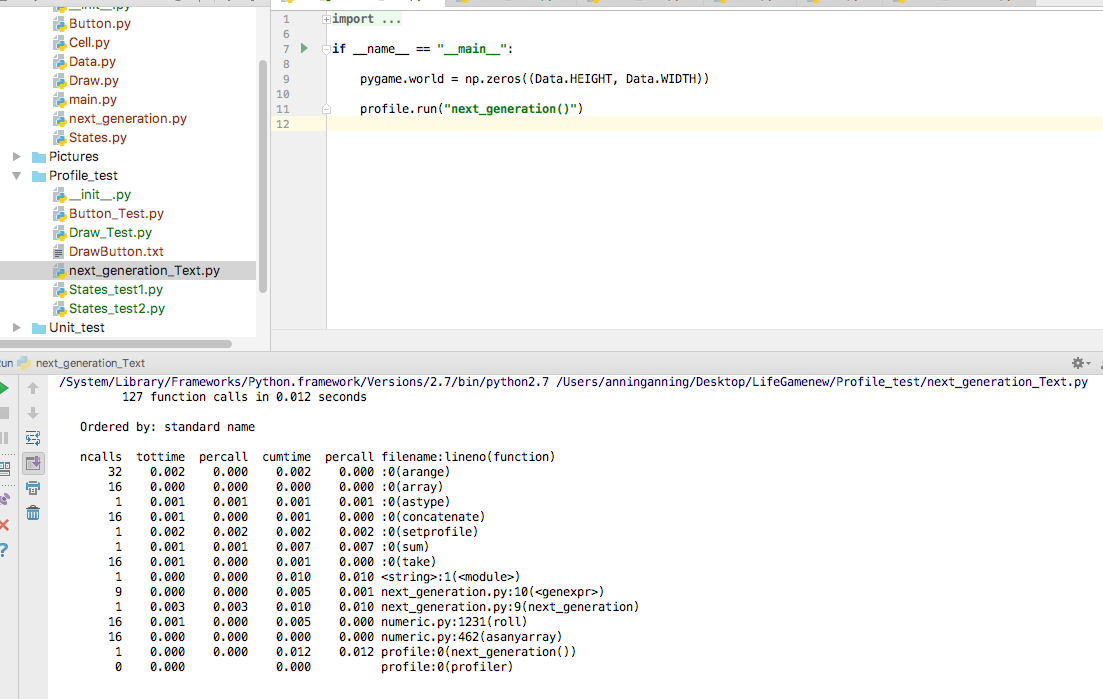


Draw.py源代码

*#-\*- coding:utf-8 –\*-***import** pygame  
**import** Data,Button  
**from** Cell **import** Cell  
  
*#按钮*button=Button.Button()  
  
*#绘制地图***def** draw(screen):  
 screen.fill((0,0,0)) *#将画布重置擦除* button.drawbutton(screen) *#画上按钮  
 #画上格子* line=1  
 size1=[line\*Cell.size **for** line **in** range(Data.HEIGHT)]  
 size2=[line\*Cell.size **for** line **in** range(Data.WIDTH+1)]*#多加1是为了将右边线画上* **for** lines **in** size1:*#画横线* pygame.draw.line(screen,(54,54,54), (0,lines), (Data.WIDTH\*Cell.size,lines))  
 **for** lines **in** size2:*#画竖线* pygame.draw.line(screen,(54,54,54), (lines,0), (lines,Data.HEIGHT\*Cell.size))  
  
 *#依次遍历细胞* **for** sp\_col **in** range(pygame.world.shape[1]):  
 **for** sp\_row **in** range(pygame.world.shape[0]):  
 **if** pygame.world[sp\_row][sp\_col]:  
 new\_cell = Cell((sp\_col \* Cell.size,sp\_row \* Cell.size))  
 screen.blit(new\_cell.image,new\_cell.rect)

next\_generation测试源代码

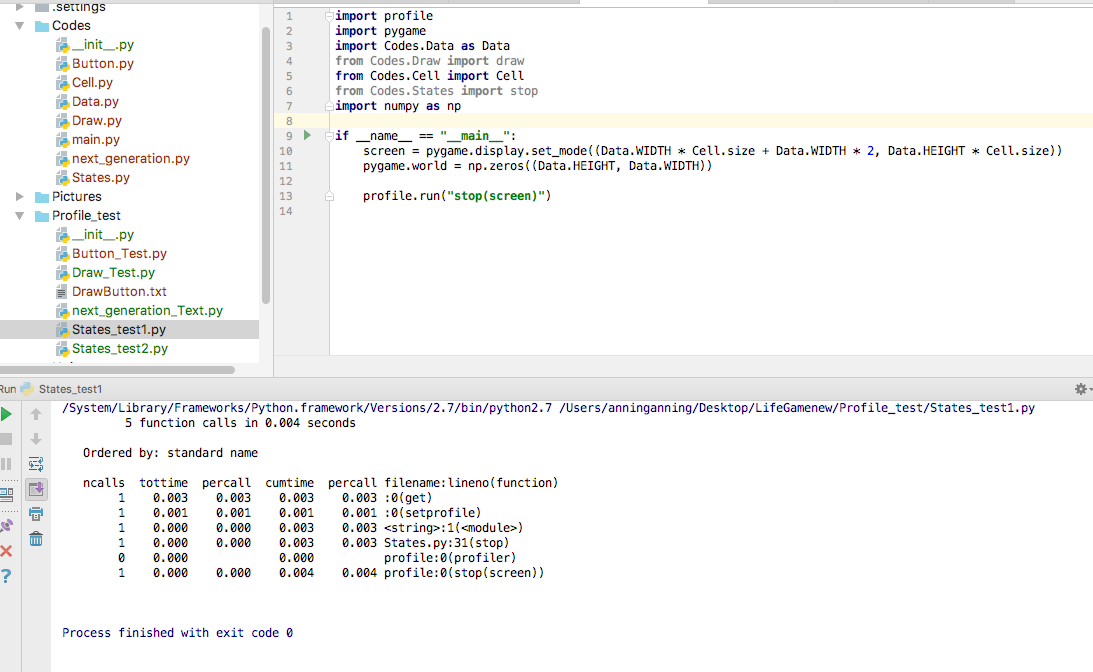
**import** profile  
**import** pygame  
**import** Codes.Data **as** Data  
**from** Codes.next\_generation **import** next\_generation  
**import** numpy **as** np  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
  
 pygame.world = np.zeros((Data.HEIGHT, Data.WIDTH))  
  
 profile.run(**"next\_generation()"**)



next\_generation.py源代码：**import** pygame  
**import** numpy **as** np  
  
  
*#根据细胞更新规则更新地图***def** next\_generation():  
 nbrs\_count = sum(np.roll(np.roll(pygame.world, i, 0), j, 1)  
 **for** i **in** (-1, 0, 1) **for** j **in** (-1, 0, 1)  
 **if** (i != 0 **or** j != 0))  
  
 pygame.world = (nbrs\_count == 3) | ((pygame.world == 1) & (nbrs\_count == 2)).astype(**'int'**)

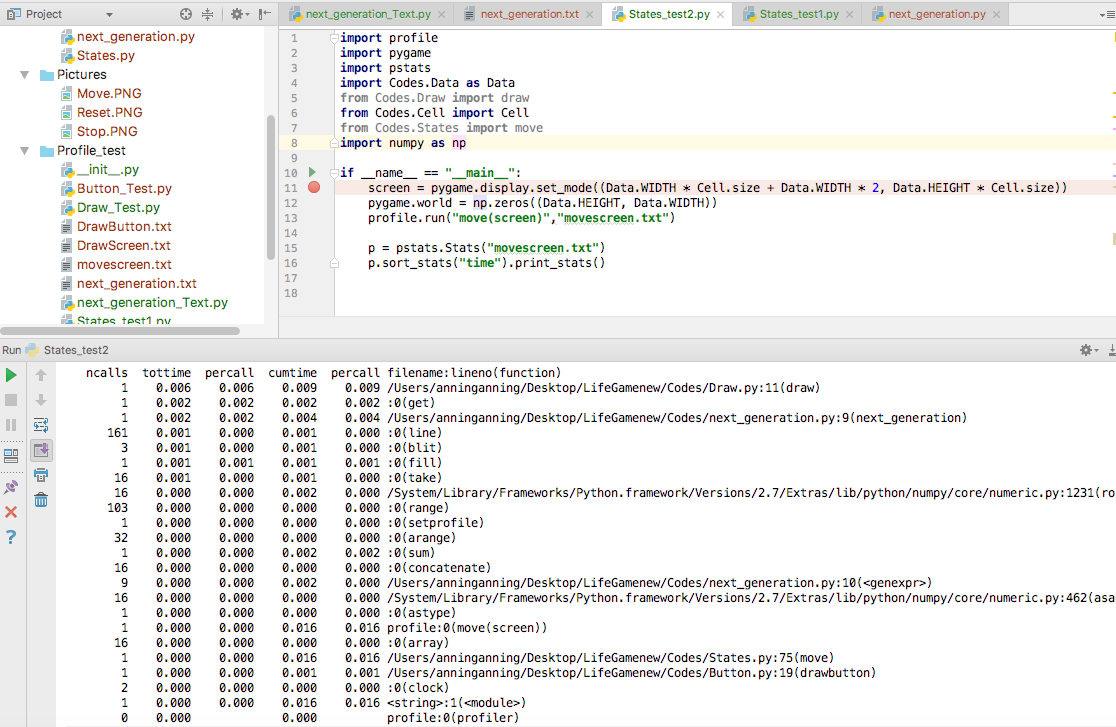
States\_Test源代码(暂停状态)

**import** profile  
**import** pygame  
**import** Codes.Data **as** Data  
**from** Codes.Draw **import** draw  
**from** Codes.Cell **import** Cell  
**from** Codes.States **import** stop  
**import** numpy **as** np  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 screen = pygame.display.set\_mode((Data.WIDTH \* Cell.size + Data.WIDTH \* 2, Data.HEIGHT \* Cell.size))  
 pygame.world = np.zeros((Data.HEIGHT, Data.WIDTH))  
  
 profile.run(**"stop(screen)"**)



states\_Test（运行状态）源代码

**import** profile  
**import** pygame  
**import** Codes.Data **as** Data  
**from** Codes.Draw **import** dra  
**from** Codes.Cell **import** Cell  
**from** Codes.States **import** move  
**import** numpy **as** np  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 screen = pygame.display.set\_mode((Data.WIDTH \* Cell.size + Data.WIDTH \* 2, Data.HEIGHT \* Cell.size))  
 pygame.world = np.zeros((Data.HEIGHT, Data.WIDTH))  
 profile.run(**"move(screen)"**)



States.py源代码：

*#-\*- coding:utf-8 –\*-***import** pygame,sys, time  
**import** numpy **as** np  
**from** pygame.locals **import** \*  
  
**import** Draw,next\_generation,Data  
**from** Cell **import** Cell  
  
*#地图状态  
  
#矩阵宽与高*WIDTH = Data.WIDTH  
HEIGHT = Data.HEIGHT  
  
*#记录鼠标按键情况的全局变量*pygame.button\_down = False  
  
*#记录游戏世界的矩阵 一个指定形状的数组*pygame.world=np.zeros((HEIGHT,WIDTH))  
  
  
*#地图初始化***def** init(screen):  
 pygame.world.fill(0) *#将数组用0填充* Draw.draw(screen)  
 **return 'Stop'***#停止时的操作***def** stop(screen):  
 *#pygame中的事件处理都放在一个while True的循环中，但是这样会造成死循环，所以在里面加一句sys.exit()来退出* **for** event **in** pygame.event.get():  
 **if** event.type == QUIT:  
 pygame.quit()  
 sys.exit()  
  
 **if** event.type == MOUSEBUTTONDOWN:*#持续按下鼠标* pygame.button\_down = True  
 pygame.button\_type = event.button  
  
 **if** event.type == MOUSEBUTTONUP:*#鼠标松开* pygame.button\_down = False  
  
 **if** pygame.button\_down:*#button\_down的bool值由上面两段确定* mouse\_x, mouse\_y = pygame.mouse.get\_pos()  
  
 sp\_col = mouse\_x / Cell.size;  
 sp\_row = mouse\_y / Cell.size;  
   
 *#画细胞* **if** sp\_col<WIDTH **and** sp\_row<HEIGHT: *#将右边的按钮栏排除在外* **if** pygame.button\_type == 1: *#鼠标左键* pygame.world[sp\_row][sp\_col] = 1  
 **elif** pygame.button\_type == 3: *#鼠标右键* pygame.world[sp\_row][sp\_col] = 0  
 Draw.draw(screen)  
   
 *#定义按钮功能  
 #鼠标已经按下，计算鼠标位置是在哪一个按钮上* **if** (2+WIDTH)\*Cell.size<mouse\_x<(2+WIDTH)\*Cell.size+Draw.button.imageMove.get\_width() **and** HEIGHT<mouse\_y<HEIGHT+Draw.button.imageMove.get\_height():  
 **return 'Move'  
 if** (2+WIDTH)\*Cell.size<mouse\_x<(2+WIDTH)\*Cell.size+Draw.button.imageStop.get\_width() **and** HEIGHT\*3<mouse\_y<HEIGHT\*3+Draw.button.imageStop.get\_height():  
 **return 'Stop'  
 if** (2+WIDTH)\*Cell.size<mouse\_x<(2+WIDTH)\*Cell.size+Draw.button.imageReset.get\_width() **and** HEIGHT\*5<mouse\_y<HEIGHT\*5+Draw.button.imageReset.get\_height():  
 **return 'Reset'  
  
 return 'Stop'***#计时器，控制帧率*pygame.clock\_start = 0  
  
  
*#进行演化时的操作***def** move(screen):  
 **for** event **in** pygame.event.get():  
 **if** event.type == QUIT:  
 pygame.quit()  
 sys.exit()  
   
 **if** event.type == MOUSEBUTTONDOWN:  
 pygame.button\_down = True  
 pygame.button\_type = event.button  
  
 **if** event.type == MOUSEBUTTONUP:  
 pygame.button\_down = False  
  
 **if** pygame.button\_down:  
 mouse\_x, mouse\_y = pygame.mouse.get\_pos() *#获得鼠标的当前位置* sp\_col = mouse\_x / Cell.size;  
 sp\_row = mouse\_y / Cell.size;  
   
 **if** sp\_col<WIDTH **and** sp\_row<HEIGHT: *#将右边的按钮栏排除在外* **if** pygame.button\_type == 1:  
 pygame.world[sp\_row][sp\_col] = 1  
 **elif** pygame.button\_type == 3:  
 pygame.world[sp\_row][sp\_col] = 0  
 Draw.draw(screen)  
   
 *#定义按钮功能* **if** (2+WIDTH)\*Cell.size<mouse\_x<(2+WIDTH)\*Cell.size+Draw.button.imageMove.get\_width() **and** HEIGHT<mouse\_y<HEIGHT+Draw.button.imageMove.get\_height():  
 **return 'Move'  
 if** (2+WIDTH)\*Cell.size<mouse\_x<(2+WIDTH)\*Cell.size+Draw.button.imageStop.get\_width() **and** HEIGHT\*3<mouse\_y<HEIGHT\*3+Draw.button.imageStop.get\_height():  
 **return 'Stop'  
 if** (2+WIDTH)\*Cell.size<mouse\_x<(2+WIDTH)\*Cell.size+Draw.button.imageReset.get\_width() **and** HEIGHT\*5<mouse\_y<HEIGHT\*5+Draw.button.imageReset.get\_height():  
 **return 'Reset'  
  
 if** time.clock() - pygame.clock\_start > 0.02:  
 next\_generation.next\_generation()  
 Draw.draw(screen)  
 pygame.clock\_start = time.clock()  
  
 **return 'Move'**