بازی دو بعدی Maze

با استفاده از کتابخانه Pygame

ارائه شده به

استاد گوران

توسط

سارا سنجابی

پروژه درس: کارگاه کامپیوتر

پاییز سال تحصیلی ۹۸-۹۷

گروه آموزشی کامپیوتر

فهرست مطالب

عنوان **صفحه**

فهرست اشکال

عنوان **صفحه**

چکیده

در این پروژه از زبان برنامه نویسی پایتون برای پیاده سازی یک بازی دوبعدی استفاده شده است. در این پروژه از کتابخانه‌ی Pygame استفاده شده است. سناریو بازی به این صورت است که اگر بازیکن بتواند از موقعیت اولیه، یعنی از گوشه بالا سمت چپ صفحه به انتهای ماز یعنی گوشه سمت راست صفحه برسد بطوری که به هیولاهای ماز برخورد نکند برنده می شود. در غیر اینصورت بازنده است. کاربر برای حرکت دادن از دکمه‌های بالا، پایین، چپ و راست روی صفحه کلید خود استفاده می کند و با استفاده از کلید Esc از برنامه خارج می‌شود.

شرح مراحل پیاده سازی

کلاس های پیاده سازی شده در این پروژه به صورت زیر است:

- کلاس Player (بازیکن)
- کلاس Maze (صفحه‌ی بازی)
- کلاس App (جهت اجرای مکانیزم بازی)

در ادامه هر یک از کلاس‌های ذکر شده به تفصیل شرح داده می‌شود.

۱- کلاس Player

این کلاس جهت پیاده سازی دو شخصیت موجود در بازی یعنی بازیکن اصلی که توسط کاربر کنترلر می شود و هیولاهای ماز که توسط خود برنامه کنترل می شود است. در این کلاس متغیرهایی مربوط به موقعیت مکانی و گام حرکت هر شی از این کلاس تعریف شده است. متغیرهای x، y متغیرهای مکانی اشیاء در صفحه‌ی Pygame هستند. متغیر های dx و dy موقعیت مکانی را در آرایه‌ی مربوط به بازی مشخص می کند. متغیر speed نیز جهت تعیین گام حرکت بازیگرها در هر حرکت است که یک متغیر ثابت است.

متدهای پیاده سازی شده در این کلاس شامل توابع حرکت به راست، حرکت به چپ، حرکت به بالا و حرکت به پایین است. این متدها توسط متد on_execute در کلاس App فراخوانی می شوند. هریک از این توابع x و y و dx و dy را تغییر می‌دهند تا موقعیت جدید هر بازیگر مشخص و ثبت شود.

تصویر کد این کلاس در شکل ۱-۱ آمده است.

```
class Player:
    x = 0
    dx = 0
    y = 44
    dy = 1
    speed = 44


    def moveRight(self):
        self.x = self.x + self.speed
        self.dx = self.dx + 1
    def moveLeft(self):
        self.x = self.x - self.speed
        self.dx = self.dx - 1

    def moveUp(self):
        self.y = self.y - self.speed
        self.dy = self.dy - 1
    def moveDown(self):
        self.y = self.y + self.speed
        self.dy = self.dy + 1
```

شکل ۱-۱: کد کلاس Player

۲- کلاس Maze

این کلاس جهت پیاده سازی صفحه‌ی بازی تعریف شده است. کد این کلاس در شکل ۱-۲ آمده است. در این کلاس متدهای تعریف شده به و کاربرد آن‌ها به شرح زیر است:

```
class Maze:
    def __init__(self):
        self.M = 12
        self.N = 12
        self.maze = [1,1,1,1,1,1,1,1,1,1,1,1,
                     0,0,0,1,0,0,0,0,1,0,1,1,
                     1,0,1,1,0,1,1,1,0,0,0,1,
                     1,0,1,0,0,0,0,0,0,1,0,1,
                     1,0,0,0,0,1,0,0,1,0,0,1,
                     1,1,0,1,1,1,0,1,0,0,1,1,
                     1,0,0,0,0,1,0,0,0,0,0,1,
                     1,0,1,1,1,1,0,0,1,1,0,0,1,
                     1,0,0,0,0,0,0,1,0,0,1,0,1,
                     1,1,0,1,0,0,0,1,0,0,0,1,
                     1,0,0,0,0,1,0,0,0,1,0,0,
                     1,1,1,1,1,1,1,1,1,1,1,1]

    def draw(self,display_surf,image_surf):
        bx = 0
        by = 0
        for i in range(0,self.M*self.N):
            if self.maze[ bx + (by*self.M) ] == 1:
                display_surf.blit(image_surf,( bx * 44 , by * 44))

            bx = bx + 1
            if bx > self.M-1:
                bx = 0
                by = by + 1
```

شکل ۱-۲: کد کلاس Maze

۲-۱- تابع __init__

متغیر های M و N تعیین کننده طول و عرض ماز که در اینجا هردو ۱۲ در نظر گرفته شده اند. در این تابع برای نمایش صفحه ی بازی و انجام حرکت‌های بازیگران از یک ماتریس ۱۲×۱۲ به نام Maze که نوع آن لیست است کمک گرفته شده است. از این متغیر در متد on_execute در کلاس App استفاده می‌شود.

۲-۲- تابع draw

این تابع به منظور ایجاد شکل گرافیکی محیط ماز در پنجره ای که هنگام اجرای برنامه برای کاربر نمایش داده می‌شود نوشته شده است. در این تابع از دو متغیر bx و by برای تعیین مختصات درج شکل بلوکهای مانعدر یک حلقه در پنجره‌ی بازی استفاده شده است.

۳- کلاس App

در واقع کلاسی که وظیفه اجرای بخش‌های مختلف بازی را بر عهده دارد. دو کلاس قبلی تنها برای ایجاد اشیاء موجود در بازی بودند. در ابتدای کلاس همانطور که انتظار می‌رود متغیرهای استفاده شده در کلاس تعریف شده اند. شرح زیر است:

۳-۱- تابع __init__

همانطور که از نام این تابع مشخص است این تابع مقادیر اولیه را برای متغیرهای موجود در کلاس مشخص می‌کند. متغیرهای مهم تعریف شده د این متد متغیرهای player، monster1، monster2 و monster3 هستند که هم اشیایی از جنس کلاس Player که در بالا توضیح داده شد هستند. این تابع فاقد مقدار ورودی و خروجی است.

۳-۲- تابع on_init

وظیفه اصلی این تابع تعریف عنوان پنجره و تعریف هریک از عکس‌ها برای استفاده در توابع مختلف کلاس است. این تابع فاقد ورودی و خروجی است.

۳-۳- تابع on_render

این تابع وظیفه‌ی اجرای گرافیکی برنامه و لود مردن تصاویر مختلف را در موقعیت‌های مختلف بازی بر عهده دارد که سه حالت متفاوت برای آن می‌توان در نظر گرفت:

۱. برنامه در حال اجرا شامل لود کردن تصاویر بازیکن و هیولاها و موانع

۲. برد بازیکن شامل نمایش تصویر برد به کاربر

۳. باخت بازیکن شامل نمایش تصویر باخت به کاربر

کد این تابع در شکل ۱-۳ آمده است.

```
def on_render(self):
    if(self.uwin == True):
        self._display_surf.blit(self._youWin_surf,(0,0))
    elif(self.ulose == True):
        self._display_surf.blit(self._youLose_surf,(0,0))

    else:
        self._display_surf.fill((255,255,255))
        self._display_surf.blit(self._image_surf,(self.player.x,self.player.y))
        self._display_surf.blit(self._monster_surf,(self.monster1.x, self.monster1.y))
        self._display_surf.blit(self._monster_surf,(self.monster2.x, self.monster2.y))
        self._display_surf.blit(self._monster_surf,(self.monster3.x, self.monster3.y))
        self.maze.draw(self._display_surf, self._block_surf)

    pygame.display.flip()
```

شکل ۱-۳: کد تابع on_render

۳-۴- تابع move_monster1، move_monster2 و move_monster3

این توابع جهت تعریف حرکت دادن خودکار هریک از هیولاها توسط برنامه نوشته شده است. به این صورت که در هربار اجرای برنامه یک عدد تصادفی بین ۰ تا ۳ انتخاب می‌شود که در واقع تعیین کننده‌ی جهت حرکت هیولاست. پس از انتخاب جهت حرکت باید مطمئن شویم که هیولا به خانه قبلی خود بر نمی‌گردد که این مقایسه با کمک دو متغیر lastx و lasty در هر تابع انجام می‌شود و همچنین باید بررسی کنیم که هیولا با حرکت خود به مانع برخورد می‌کند یا خیر. در ادامه پس از انجام حرکت، باید متغیرهای مربوطه را در شئ مورد نظر آپدیت کنیم.

یک نمونه از این توابع در شکل۲-۳ آمده است.

```
def move_monster1(self):
    lastx = self.monster1.dx
    lasty = self.monster1.dy
    move = randint(0, 4)
    if(move == 0 and
       self.maze.maze[self.monster1.dx+(12*(self.monster1.dy))+1]!=1 and
       lastx != self.monster1.dx+1):

        lastx = self.monster1.dx
        lasty = self.monster1.dy
        self.monster1.moveRight()
    elif(move == 1 and
       self.maze.maze[self.monster1.dx+(12*(self.monster1.dy))-1]!=1 and
       lastx != self.monster1.dx-1):

        lastx = self.monster1.dx
        lasty = self.monster1.dy
        self.monster1.moveLeft()
    if(move == 2 and
       self.maze.maze[self.monster1.dx+(12*(self.monster1.dy-1))]!=1 and
       lastx != self.monster1.dy-1):

        lastx = self.monster1.dx
        lasty = self.monster1.dy
        self.monster1.moveUp()
    if(move == 3 and
       self.maze.maze[self.monster1.dx+(12*(self.monster1.dy+1))]!=1 and
       lastx != self.monster1.dy+1):

        lastx = self.monster1.dx
        lasty = self.monster1.dy
        self.monster1.moveDown()
```

شکل ۲-۳: کد تابع move_monster1

۳-۵- تابع on_execute

در این تابع ابتدا تابع ()on_init از همین کلاس فراخوانی می‌شود. سپس یک حلقه‌ی while بی‌نهایت اجرا می‌شود تا بررسی شود که کاربر چه کلیدی را برای حرکت دادن بازیکن فشار داده است. پس از آن بررسی می شود که اگر بازیکن به هریک از هیولاها برخورد کرده باشد، پیام "شما باختید" به کاربر نمایش داده شود و اگر بازیکن به خانه انتهایی ماز رسیده است پیام "شما بردید" به کاربر نمایش داده شود. همچنین در انتهای حلقه هیولاهای بازی توسط توابع move_monster1، move_monster2 و move_monster3 حرکت داده می شوند. در انتهای حلقه نیز تابع ()on_render از همین کلاس فراخوانی می‌شود. کد این تابع در شکل ۳-۳ نشان داده شده است.

```
def on_execute(self):
    if self.on_init() == False:
        self._running = False

    while( self._running ):
        time.sleep(0.2)
        pygame.event.pump()
        keys = pygame.key.get_pressed()

        if (keys[K_RIGHT]):
            if (self.maze.maze[self.player.dx+(12*(self.player.dy))+1]!=1):
                self.player.moveRight()

        if (keys[K_LEFT]):
            if (self.maze.maze[self.player.dx+(12*(self.player.dy))-1]!=1):
                self.player.moveLeft()

        if (keys[K_UP]):
            if (self.maze.maze[self.player.dx+(12*(self.player.dy-1))]!=1):
                self.player.moveUp()

        if (keys[K_DOWN]):
            if (self.maze.maze[self.player.dx+(12*(self.player.dy+1))]!=1):
                self.player.moveDown()

        if (keys[K_ESCAPE]):
            self._running = False

        if(self.player.dx == 11 and self.player.dy == 10):
            self.uwin = True

        if(self.player.dx == self.monster1.dx and self.player.dy == self.monster1.dy):
            self.ulose = True
        if(self.player.dx == self.monster2.dx and self.player.dy == self.monster2.dy):
            self.ulose = True
        if(self.player.dx == self.monster3.dx and self.player.dy == self.monster3.dy):
            self.ulose = True

        self.move_monster1()
        self.move_monster2()
        self.move_monster3()
        self.on_render()
```

شکل ۳-۳: کد تابع on_execute

پیوست ها

پیوست ۱

از قطعه کد زیر در این برنامه استفاده شده است:

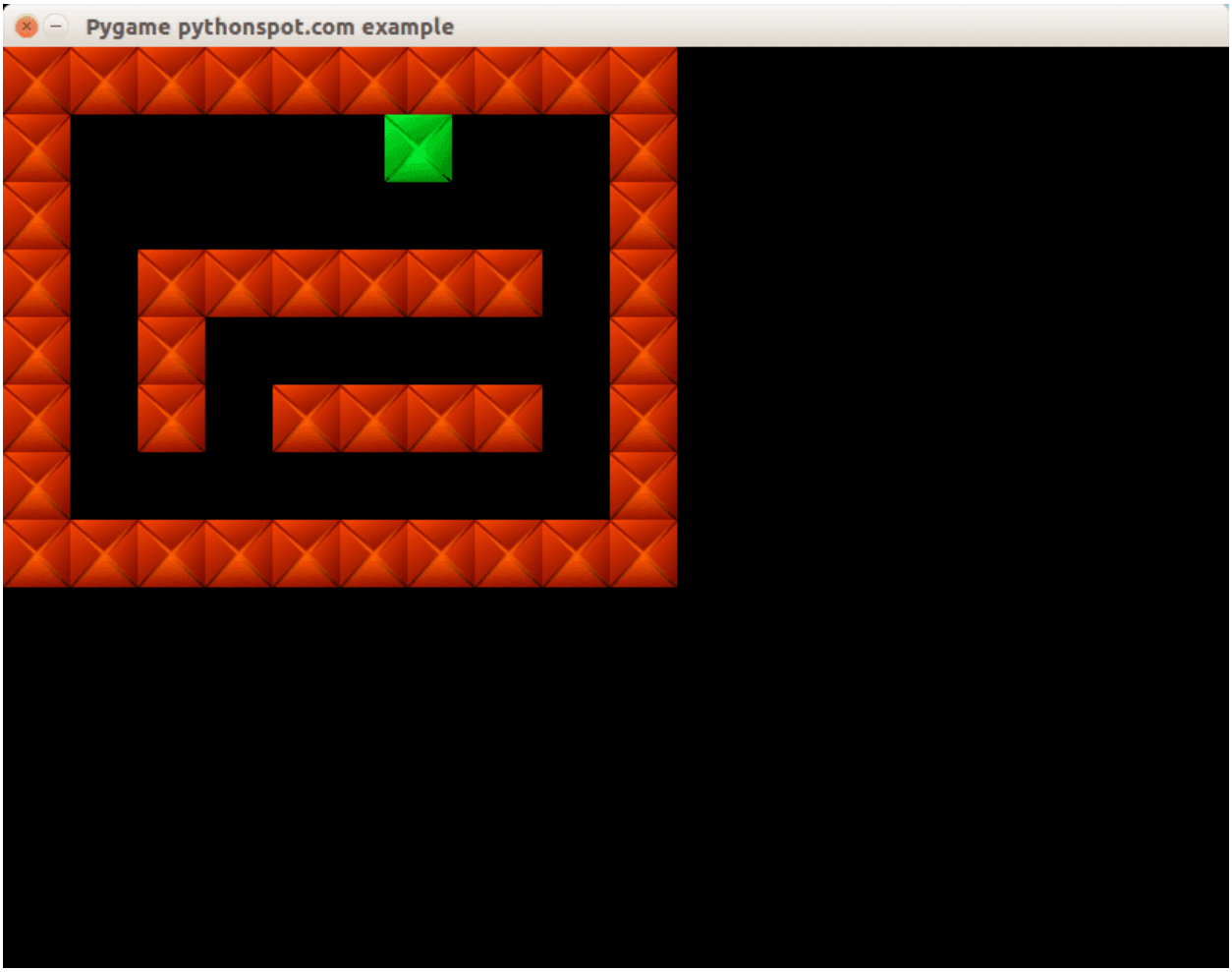
منبع شماره دو نشان دهنده ی مرجع این کد می باشد.

```
from pygame.locals import *
import pygame

class Player:
    x = 44
    y = 44
    speed = 1

    def moveRight(self):
        self.x = self.x + self.speed

    def moveLeft(self):
        self.x = self.x - self.speed

    def moveUp(self):
        self.y = self.y - self.speed

    def moveDown(self):
        self.y = self.y + self.speed

class Maze:
    def __init__(self):
       self.M = 10
       self.N = 8
       self.maze = [ 1,1,1,1,1,1,1,1,1,1,
                     1,0,0,0,0,0,0,0,0,1,
                     1,0,0,0,0,0,0,0,0,1,
                     1,0,1,1,1,1,1,1,0,1,
                     1,0,1,0,0,0,0,0,0,1,
                     1,0,1,0,1,1,1,1,0,1,
                     1,0,0,0,0,0,0,0,0,1,
                     1,1,1,1,1,1,1,1,1,1,]
```

```
    def draw(self,display_surf,image_surf):
        bx = 0
        by = 0
        for i in range(0,self.M*self.N):
            if self.maze[ bx + (by*self.M) ] == 1:
                display_surf.blit(image_surf,( bx * 44 ,
by * 44))

            bx = bx + 1
            if bx > self.M-1:
                bx = 0
                by = by + 1


class App:

    windowWidth = 800
    windowHeight = 600
    player = 0

    def __init__(self):
        self._running = True
        self._display_surf = None
        self._image_surf = None
        self._block_surf = None
        self.player = Player()
        self.maze = Maze()

    def on_init(self):
        pygame.init()
        self._display_surf =
pygame.display.set_mode((self.windowWidth,self.windowHe
ight), pygame.HWSURFACE)

        pygame.display.set_caption('Pygame
pythonspot.com example')
        self._running = True
        self._image_surf =
pygame.image.load("player.png").convert()
```

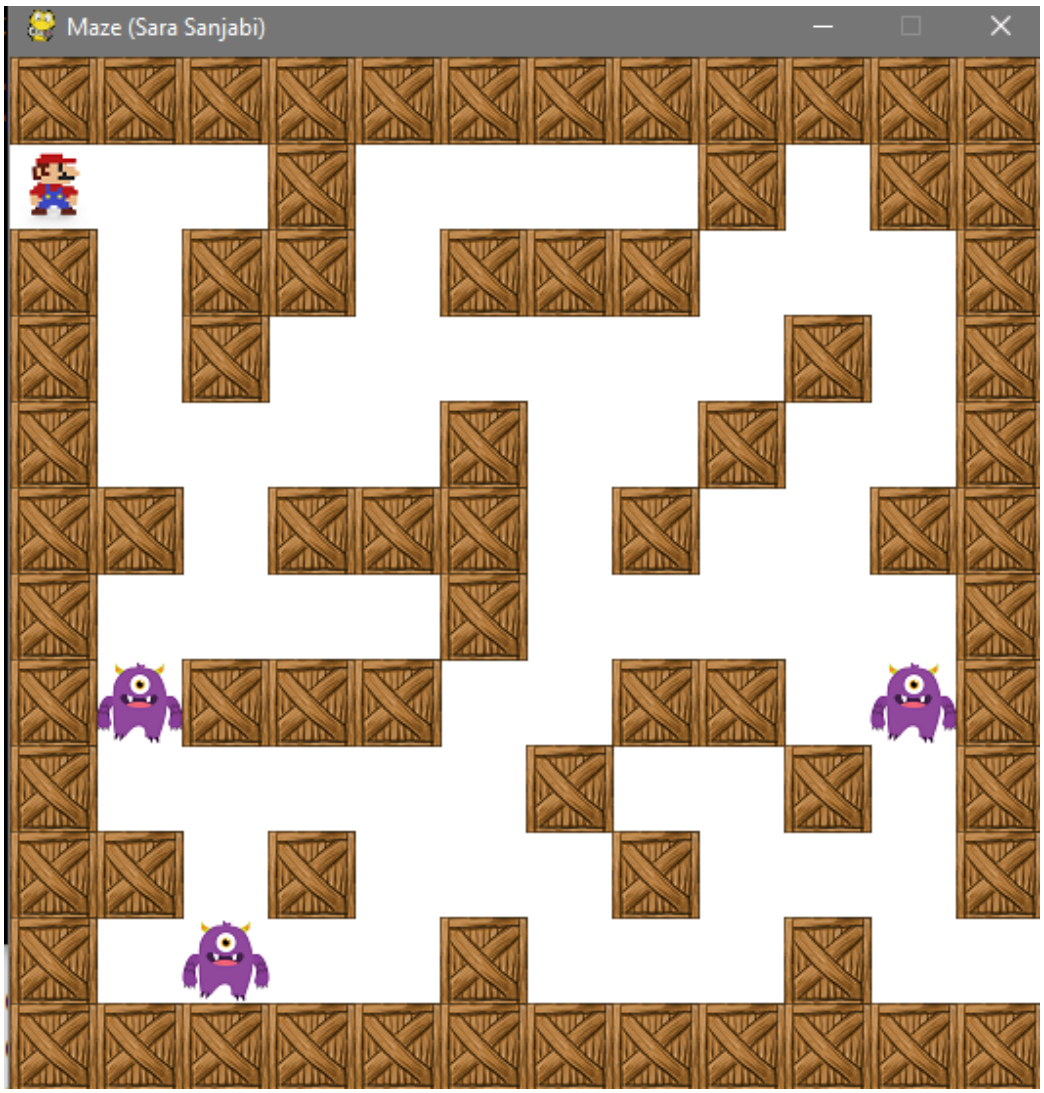
```
        self._block_surf =
pygame.image.load("block.png").convert()

    def on_event(self, event):
        if event.type == QUIT:
            self._running = False

    def on_loop(self):
        pass

    def on_render(self):
        self._display_surf.fill((0,0,0))

self._display_surf.blit(self._image_surf,(self.player.x
,self.player.y))
        self.maze.draw(self._display_surf,
self._block_surf)
        pygame.display.flip()

    def on_cleanup(self):
        pygame.quit()

    def on_execute(self):
        if self.on_init() == False:
            self._running = False

        while( self._running ):
            pygame.event.pump()
            keys = pygame.key.get_pressed()

            if (keys[K_RIGHT]):
                self.player.moveRight()

            if (keys[K_LEFT]):
                self.player.moveLeft()

            if (keys[K_UP]):
                self.player.moveUp()

            if (keys[K_DOWN]):
```

```
                self.player.moveDown()

            if (keys[K_ESCAPE]):
                self._running = False

            self.on_loop()
            self.on_render()
        self.on_cleanup()

if __name__ == "__main__" :
    theApp = App()
    theApp.on_execute()
```



پیوست ۲

تصویری از اجرای برنامه به صورت زیر است:



پیوست ۳

لینک گیت هاب پروژه:

https://github.com/elsan96/project/

منابع:

1. https://stackoverflow.com/
2. https://pythonspot.com/maze-in-pygame/