قسمت نهم

این مار خوش خط‌وخال

توسعه بازی‌های ساده با پایتون

» احمد شریف‌پور

بازی‌های دیجیتال از ابتدای صنعت محاسبات کامپیوتری یکی از سودآورترین و پرطرفدارترین بازوهای این صنعت محسوب می‌شده‌اند. از دیگر سو، برنامه‌نویسی بازی‌های کامپیوتری به دلیل تعامل شدید با کاربر و همچنین به لحاظ نیاز به واکنش‌های سریع از سوی سیستم و در مواردی شبیه‌سازی هوش و رفتار انسانی از دشوارترین مواردی است که یک برنامه‌نویس می‌تواند در آن مهارت پیدا کند. اما این دشواری سبب نخواهد شد که ما این بخش را به‌صورت کامل کنار بگذاریم. در این قسمت به طراحی و ساخت یک بازی بسیار ساده خواهیم پرداخت. در واقع برنامه‌ای خواهیم نوشت که یک کاراکتر دلخواه را روی صفحه بازی به فرمان کاربر جابه‌جا می‌کند. همانند بازی‌های پرطرفدار و مشهور که بر اساس یک موتور بازی توسعه داده می‌شوند، ما نیز در این بخش از مجموعه مقالات پایتون به بررسی یکی از ماجول‌های مخصوص توسعه بازی یعنی PyGame خواهیم پرداخت. هرچند برنامه ما عملاً یک «بازی» نخواهد بود، اما شما را با اصول اولیه کار آشنا خواهد کرد.

در ایـن قسـمت قصد داریم بـه بررسـی Pygame بپردازیم. درواقع Pygame مجموعـه‌ای از ماجول‌هـا اسـت کـه بـا هـدف ساده‌سـازی روند توسـعه بازی‌های سـاده طراحی‌شـده‌اند. این ماجول‌ها از سـایت www.pygame.org قابـل دریافت هسـتند. به گفته توسـعه‌دهندگان این مجموعه: «Pygame کتابخانه‌ای مستقل از پلتفرم است که برای ساده‌سازی توسـعه نرم‌افزارهای مالتی‌مدیا نظیر، بازی‌ها، توسط زبان پایتون طراحی شـده اسـت. Pygame برای اجرا به زبان برنامه‌نویسی پایتون و کتابخانه مالتی‌مدیـای SDL نیـاز دارد.» بـرای نصـب Pygame روی اوبونتـو به سـادگی می‌توانید با مراجعه به Synaptic بسـته نرم‌افـزاری آن را با نام python-pygame دانلود کنید یا در ترمینال دستور زیر را وارد کنید:

```
sudo apt-get install python-pygame
```

صفحه بازی

برای شروع کار، فهرست ۱ را در ویرایشگر متن دلخواهتان وارد کنید و بـا نـام game1.py آن را ذخیره کنید. در خط ۳ ایـن برنامه ما Pygame را import می‌کنیم تا توابع و مقادیر آن را در اختیار داشـته باشـیم. پس از آن و در خـط ۵ ماجـول os را وارد می‌کنیـم. هدف اصلی از به‌کار بردن ماجول os در خط ۶ مشـخص می‌شـود. جایی که ما با مقداردهی یکی از مقادیر متغیر environment از ماجول os (که از جنس لیسـت است) تعیین می‌کنیم پنجره مربوط به برنامه بازی ما درسـت در وسـط صفحه نمایش بـاز شـود. پس از آن در خط ۸ کـد Pygame را راه‌انـدازی یا به اصطلاح initialize می‌کنیم. این کار باعث می‌شـود، پنجره مربوط به بازی ما تولید شـده و در وسـط صفحه به نمایش درآید. در مرحله بعد و در خط شماره ۱۰ انـدازه ایـن پنجره را برابـر ۸۰۰×۶۰۰ و در خط ۱۱ عنـوان پنجره را برابـر عبـارت "Pygame Test 1" تنظیم می‌کنیـم. در نهایت، با اجرای یک حلقه به انتظار یک رویداد (در اینجا فشـرده شدن کلیدهای ماوس یا کلیدی از صفحـه کلید) می‌نشـینیم. نخسـتین برنامه‌ای که ما بـا کمک Pygame نوشـته‌ایم، هیـچ عملیات خاصی انجام نمی‌دهد و تنهـا هدف ما از آوردن این مثال آشـنایی با روش راه‌انـدازی و تنظیم خصوصیات ابتدایی پنجره برنامـه نظیـر اندازه و عنـوان پنجره بود. آنچه در این‌جا مهم اسـت درک شـیء screen به عنـوان یک «ظرف» یـا Container اسـت. در اصطلاح

```
1  #! /usr/bin/env python
2  # importing modules
3  import pygame
4  from pygame.locals import *
5  import os
6  os.environ[‹SDL_VIDEO_CENTERED›]=›1›
7  # Initializing the game
8  pygame.init()
9  # screen setup
10 screen = pygame.display.set_mode((800,600))
11 pygame.display.set_caption(‹Pygame Test 1›)
12 doloop=1
13 while doloop:
14     if pygame.event.wait().type in (KEYDOWN ,\
15 MOUSEBUTTONDOWN):
16         break
```

فهرست۱ کد ساده‌ای برای ایجاد یک صفحه Pygame

Pygame به آن سطح یا Surface گفته می‌شود. این شیء به نوعی نقش یک برگه کاغذ را بازی خواهد کرد که ما ترسـیمات مورد نظرمان را روی آن انجام می‌دهیم.

اکنـون می‌توانیـم بـه تدریج قابلیت‌هـا و تنظیماتی را بـه این برنامه بیافزاییـم. برای شـروع از تغییر پس زمینه برنامه شـروع می‌کنیم. برای تنظیـم رنگ در ماجول‌های Pygame مجبور هسـتیم مقادیر RGB یا قرمز، سـبز و آبی رنگ را تعیین کنیم. برای سـاده‌تر شـدن این کار می‌توانید از نرم‌افـزاری به نام colorname اسـتفاده کنید کـه از طریق مرکز نرم‌افزار اوبونتـو قابل نصب اسـت. به کمـک این نرم‌افزار تعییـن مقادیر RGB هر رنـگ دلخـواه از طریق یک چرخ رنگ به سـادگی امکان‌پذیر خواهد شـد. برای تنظیم رنگ پس‌زمینه کافی اسـت، پس از دسـتور import خط زیر را اضافه کنید:

```
Background = 208 , 202 , 104
```

این کار رنگی آجری را به متغیر Background (که از جنس توپل است) نسبت خواهد داد. آن‌گاه پس از دستور pygame.display.set_caption دو خط زیر را اضافه کنید:

```
screen.fill(Background)
screen.display.update()
```

متـد screen.fill() تنظیـم رنگ صفحه را بر عهده دارد و متد update() باعث اعمال تنظیم انجام شده به صفحه در حال نمایش خواهد شد.

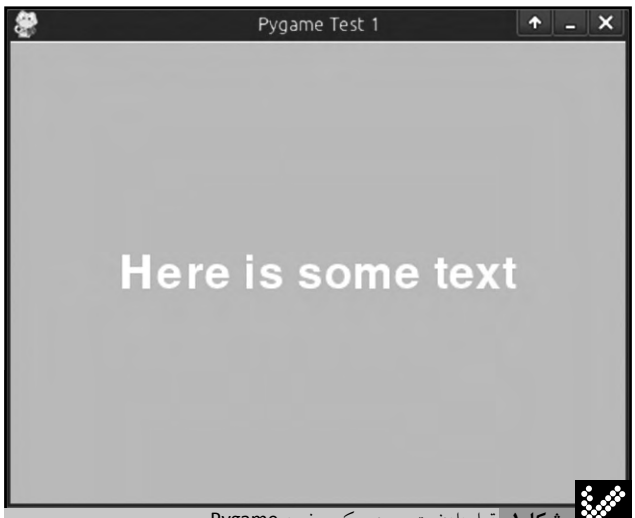
نمایش متن

اکنون زمان آن رسیده است که چند سطر متن را به پنجره برنامه اضافه کنیم. درسـت بعد از تعریف رنگ پس‌زمینه دستور زیر را برای تنظیم رنگ پیش‌زمینه به سفید وارد کنید:

```
FontForeground = 255 , 255 , 255
```

اگـر برنامه را در این وضعیت اجرا کنید، اتفاق خاصی رخ نخواهد داد، زیـرا ما تاکنون تنهـا رنگ متن را تعیین کرده‌ایم. برای نوشـتن متن روی screen یا همان پنجره برنامه، خطوط زیر را بین دو دستور screen.fill() و pygame.display.update() وارد کنید.

```
font = pygame.font.Font(None , 27)
text = font.render('Here is some text' ,
True , FontForeground , Background)
textrect = text.get_rect()
screen.blit(text,textrect)
```



شکل۱ قراردادن متن روی یک صفحه Pygame

این برنامه را با عنوان game2.py ذخیره کرده و اجرا کنید. برنامه شما اکنون باید شبیه فهرست ۲ باشد. برای نوشتن متن ما ابتدا در خط ۱۵ متد Font را فراخوانده و ۲ آرگومان را به آن پاس کرده‌ایم. نخستین آرگومان نام فونت و دومی اندازه آن خواهد بود. در اینجا ما از None برای نام فونت استفاده کرده‌ایم که باعث می‌شود سیستم از فونت پیش‌فرض خود استفاده کند.

پس از آن و در خط ۱۶ متد ()font.render را به‌کار برده‌ایم که چهار آرگومان دارد. این چهار آرگومان به ترتیب عبارتند از متن موردنظر، استفاده یا عدم استفاده از anti-aliasing، رنگ پیش‌زمینه متن و در نهایت رنگ پس‌زمینه آن. برای قرار دادن متن روی screen از واسطی به نام مستطیل یا Rectangle استفاده می‌کنیم. در خط ۱۸ با استفاده از تابع ()text.get_rect مستطیل مربوط به متن را ایجاد کرده‌ایم. درک این روند بسیار مهم است، زیرا اغلب اشیایی که از این به بعد با آن‌ها سروکار خواهیم داشت، مستطیل خواهند بود.

پس از آن ما مستطیل حاوی متن را روی screen قرار داده (خط ۱۹) و تغییرات را اعمال کرده‌ایم. اما blit چیست و چرا از چنین نام عجیب و غریبی استفاده می‌شود؟ ریشه‌های این نام به سال‌های ۱۹۷۰ و دوران زیراکس پارک (جایی که بسیاری از تکنولوژی‌های امروزی را مدیون دانشمندان و نوآوران فعال در آن هستیم) باز می‌گردد. این اصطلاح در واقع خلاصه Bitmap Block Transfer است. وظیفه این دستور قرار دادن یک بلوک تصویری روی یک پس‌زمینه است.

```
1  #! /usr/bin/env python
2  # importing modules
3  import pygame
4  from pygame.locals import *
5  import os
6  os.environ[‹SDL_VIDEO_CENTERED›]=›1›
7  Background = 208,202,104
8  FontForeground = 255,255,255
9  # Initializing the game
10 pygame.init()
11 # screen setup
12 screen = pygame.display.set_mode((800,600))
13 pygame.display.set_caption(‹Pygame Test 2›)
14 screen.fill(Background)
15 font=pygame.font.Font(None,27)
16 text=font.render(‹Here is some text›,True,\
17 FontForeground,Background)
18 textrect=text.get_rect()
19 screen.blit(text,textrect)
20 pygame.display.update()
21 doloop=1
22 while doloop:
23     if pygame.event.wait().type in (KEYDOWN ,\
24 MOUSEBUTTONDOWN):
25         break
```

فهرست ۲

اما اگر بخواهیم متن ما درست در وسط پنجره برنامه ظاهر شود، چه باید کرد؟ در چنین صورتی کافی است بین دستور ()text.get_rect (خط ۱۸) و screen.blit (خط ۱۹) این دو خط را اضافه کنید:

```
textRect.centerx = screen.get_rect().centerx
textRect.centery = screen.get_rect().centery
```

همان‌گونه که مشاهده می‌کنید، خود screen هم در واقع یک مستطیل است که ما به کمک متد ()get_rect مشخصات آن و به ویژه x و y مرکز آن را استخراج کرده‌ایم و آن‌ها را به مختصات مرکز مستطیل متن نسبت داده‌ایم.

دوباره برنامه را اجرا کنید. این بار مشاهده خواهید کرد که همانند شکل ۱ پس از اجرای آن، متن Here is some text درست در وسط پنجره برنامه ظاهر می‌شود. با استفاده از متدهای set_bold(True) و set_italic(True) درست پس از قسمت pygame.font.Font می‌توانید به متن حالت توپر یا مایل را نیز نسبت دهید. اگر به خاطر داشته باشید، پیش‌تر گفتیم که برای تنظیم فونت متن از متد ()pygame.font.Font استفاده می‌کنیم که دو آرگومان فونت و اندازه را دریافت می‌کند. مشکل اینجا است که ما نمی‌توانیم از محل قرارگیری و نصب فونت‌های مختلف روی سیستم‌های متفاوت اطلاع پیدا کنیم یا درباره حالتی خاص مطمئن باشیم. خوشبختانه pygame این امکان را با پیش‌بینی متدی به نام match_font حل کرده است. در فهرست ۳ شما قطعه کدی را مشاهده می‌کنید که می‌تواند به عنوان مثال، محل قرارگیری فونت Courier New را برای ما چاپ کند. این کد در سیستم نگارنده آدرس usr/share/fonts/truetype/freefont/FreeMono.ttf را بازمی‌گرداند که ممکن است در سیستم شما مقدار دیگری باشد. اگر فونت Courier New روی سیستم نصب نشده باشد، این کد مقدار None را بازخواهد گرداند. اما اگر فرض کنیم این فونت روی سیستم نصب شده باشد، می‌توان مقدار برگشتی این تابع را مانند قطعه کد زیر برای تنظیم فونت برنامه به کار برد:

```
courier = pygame.font.match_font('Courier New')
font = pygame.font.Font(courier , 27)
```

این دو خط را نیز می‌توانید به عنوان آخرین تغییرات به کد برنامه game3.py خود اعمال کنید. در هنگام ایجاد یک بازی واقعی نهایت کار آن است که از یک فونت مشخص که مطمئن هستید روی سیستم‌های هدف موجود است، استفاده کنید یا فونت موردنظر را به همراه بسته نصب بازی عرضه کنید.

گرافیک

اگرچه متن‌ها زیبا هستند، اما گرافیک‌ها از آن‌ها زیباترند. در اینجا به توضیح نحوه کار اسپرایت‌ها (Sprite) در pygame خواهیم پرداخت. اسپرایت‌ها تصاویر دو بعدی ساده‌ای هستند که در نرم‌افزارهای گرافیکی برای نمایش اجزای بازی مورد استفاده قرار می‌گیرند. برای استفاده از گرافیک‌ها در برنامه بازی خودتان، توسط نرم‌افزاری نظیر GIMP یا ابزارهای مشابه تصویری شبیه یک آدمک را در یک فایل ۵۰ × ۵۰ پیکسل ترسیم کنید. رنگ پس‌زمینه تصویر را به حالت شفاف (Transparent) تغییر دهید و فایل را با نام stick.png در پوشه‌ای که فایل‌های کد برنامه در آن قرار دارند، ذخیره کنید. توجه کنید که حتماً حالت پس‌زمینه تصویر را روی Transparent تنظیم کنید تا هنگام نمایش و حرکت اسپرایت روی پس‌زمینه تنها تصویر آدمک دیده شود، نه یک مربع ۵۰ × ۵۰ سفید رنگ.

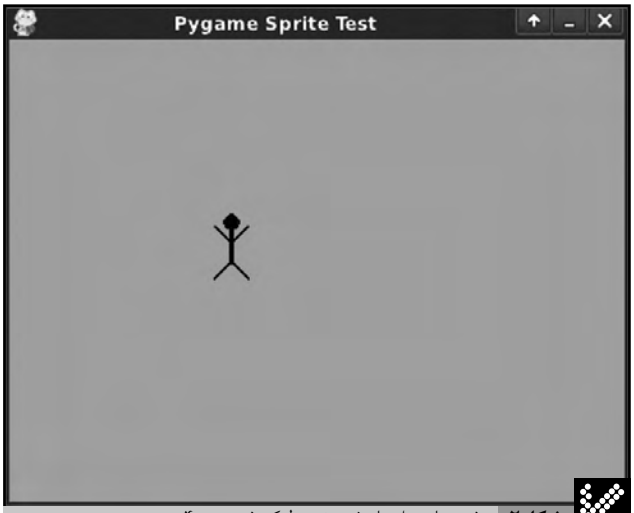
```
import pygame
from pygame.locals import *
print pygame.font.match_font(‹Courier New›)
```

فهرست ۳ کدی برای تعیین محل یک فونت خاص

هدف بعدی ما، طراحی بازی به گونه‌ای است که این اسپرایت روی صفحه بازی نمایش داده شود و با فشرده شدن کلیدهای جهت‌نما، آدمک نیز به چپ و راست و بالا و پایین حرکت کند. این حرکت باید در لبه‌های صفحه بازی متوقف شود و با فشرده شدن کلید q بازی خاتمه یابد. حرکت دادن اسپرایت روی صفحه به نسبت ساده است. تنها باید بدانید که برای این کار از دو اسپرایت؛ یکی

```
1  #! /usr/bin/env python
2  # importing modules
3  import pygame
4  from pygame.locals import *
5  import os
6  import sys
7  os.environ[‹SDL_VIDEO_CENTERED›]=›1›
8  Background = 0,255,127
9  FontForeground = 255,255,255
10 # Initializing the game
11 pygame.init()
12 # screen setup
13 screen = pygame.display.set_mode((400,300))
14 pygame.display.set_caption(‹Pygame Sprite Test›)
15 screen.fill(Background)
16 class Sprite(pygame.sprite.Sprite):
17     def __init__(self,position):
18         pygame.sprite.Sprite.__init__(self)
19         # get the screen rectangle
20         self.screen = pygame.display.get_surface().get_rect()
21         # Variable to store privious position
22         self.oldsprite=(0,0,0,0)
23         self.image=pygame.image.load(‹stick.png›)
24         self.rect=self.image.get_rect()
25         self.rect.x=position[0]
26         self.rect.y=position[1]
27      def update(self,amount):
28         # make a copy of current rectangle
29         self.oldsprite = self.rect
30         # Moving the rectangle
31         self.rect = self.rect.move(amount)
32         # Check the border
33         if self.rect.x <0:
34             self.rect.x=0
35         elif self.rect.x > (self.screen.width - self.rect.width):
36             self.rect.x = self.screen.width - self.rect.width
37         if self.rect.y < 0:
38             self.rect.y=0
39         elif self.rect.y > (self.screen.height - self.rect.height):
40             self.rect.y = self.screen.height - self.rect.height
41 character = Sprite((screen.get_rect().x , screen.get_rect().y))
42 screen.blit(character.image,character.rect)
43 blank=pygame.Surface((character.rect.width,character.rect.height))
44 blank.fill(Background)
45 pygame.display.update()
46 doloop=1
47 while doloop:
48     for event in pygame.event.get():
49         if event.type == pygame.QUIT:
50             sys.exit()
51         elif event.type == pygame.KEYDOWN:
52             if event.key == pygame.K_LEFT:
53                 character.update([-10,0])
54             elif event.key == pygame.K_UP:
55                 character.update([0,-10])
56             elif event.key == pygame.K_RIGHT:
57                 character.update([10,0])
58             elif event.key == pygame.K_DOWN:
59                 character.update([0,10])
60             elif event.key == pygame.K_q:
61                 doloop=0
62         # Erase old position
63         screen.blit(blank,character.oldsprite)
64         # Draw new
65         screen.blit(character.image,character.rect)
66         # Update modified positions
67         pygame.display.update([character.oldsprite,character.rect])
```

فهرست ۴ حرکت دادن یک کاراکتر روی صفحه Pygame



شکل ۲ صفحه بازی ایجاد شده توسط کد فهرست ۴

اسپرایت آدمک و دیگری یک اسپرایت خالی استفاده خواهیم کرد. زمانی که کلیدی فشرده می‌شود، برنامه موقعیت جدید اسپرایت آدمک را محاسبه می‌کند، سپس ما اسپرایت خالی را در محل فعلی آدمک قرار می‌دهیم تا به اصطلاح آن را پاک کنیم، آن گاه اسپرایت آدمک را در محل جدید ظاهر می‌کنیم. یک فایل جدید ایجاد کنید و محتویات فهرست ۴ را در آن وارد کرده و آن را با نام game4.py ذخیره کنید. با اجرای این کد شما آدمک متحرک را روی پس‌زمینه‌ای به رنگ سبز تیره مشاهده خواهید کرد که به کلیدهای جهت‌نما واکنش نشان داده و به دستور شما حرکت خواهد کرد. بخش اول این کد، شبیه مثال‌های قبلی و شامل خطوطی برای import ماجول‌های موردنیاز و تنظیم رنگ و عنوان صفحه بازی است. پس از آن در خط ۱۶ کلاسی را برای کنترل و مدیریت اسپرایت یا گرافیک دلخواهمان تعریف کرده‌ایم. در ابتدای کلاس و با تعریف تابع __init__ ما اسپرایت را راه‌اندازی می‌کنیم. غالب کارهای ساخت اسپرایت به کمک متد اصلی __pygame.sprite.Sprite.__init که در خود pygame تعریف شده است (خط ۱۸) به انجام می‌رسد. پس از آن در خط ۲۰ سطح یا Surface بازی را تعریف کرده و آن را screen نامیده‌ایم. بعدها به کمک شیء screen می‌توانیم کنترل کنیم که اسپرایت از صفحه بازی خارج نشود. سپس در خط ۲۳ با تابع pygame.image.load فایل تصویر آدمک ساخته شده را به اسپرایت نسبت داده‌ایم. اگر فایل برنامه و فایل تصویر در یک پوشه نباشند، باید آدرس کامل فایل تصویری در این تابع قید شود. پس از آن در خط‌های ۲۵ و ۲۶ موقعیت x, y اسپرایت توسط متغیر position که به تابع پاس شده است، تنظیم می‌شود.

تابع update که در خط ۲۷ تعریف شده است، وظیفه حرکت دادن اسپرایت و پرکردن محل قبلی آن با یک فضای خالی (پاک کردن اسپرایت) را برعهده خواهد داشت. این تابع همچنین عدم خروج اسپرایت از صفحه را کنترل می‌کند. پس از پایان تعریف کلاس، ما نمونه‌ای از کلاس را در خط ۴۱ با نام character ایجاد کرده و با تابع blit روی صفحه نمایش داده‌ایم (خط ۴۲). پس از آن شیء blank یا پاک کننده ایجاد شده و تابع update ماجول display به اجرا در می‌آید. پس از آن در خطوط ۴۷ تا ۶۷ یک حلقه بی‌پایان ایجاد شده و با کنترل ورودی کاربر، آدمک ساخته شده روی صفحه نمایش به حرکت در خواهد آمد. کارهای بسیار پیچیده‌تری با pygame قابل انجام است به عنوان مثال، این ماجول دارای توابعی برای کنترل برخورد اسپرایت‌ها (مثلاً برخورد گلوله با شخصیت بازی)، نگه‌داری امتیازها و افزودن جلوه‌های صوتی و... نیز دارد. اما هدف ما آشناکردن شما با مبانی کار با این ماجول بود و امیدواریم که توانسته باشیم حس کنجکاوی شما را برای آگاهی بیشتر برانگیخته باشیم.