



بازی سازی (pygame)

پایتون برای کودکان و نوجوانان

Python FOR KIDS & TEENS

آزمایشگاه بهینه سازی

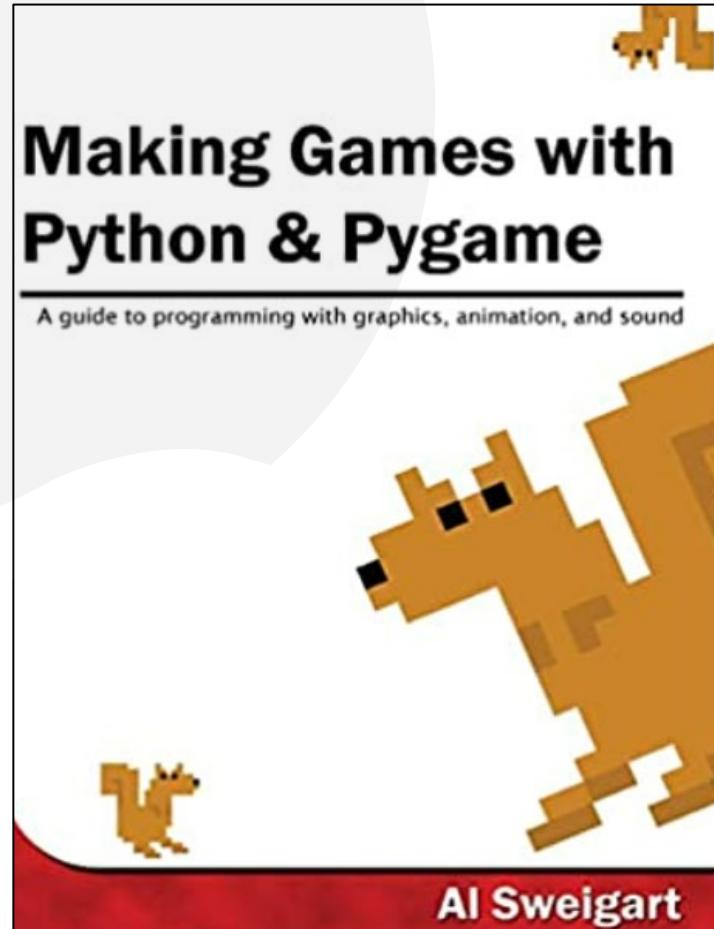
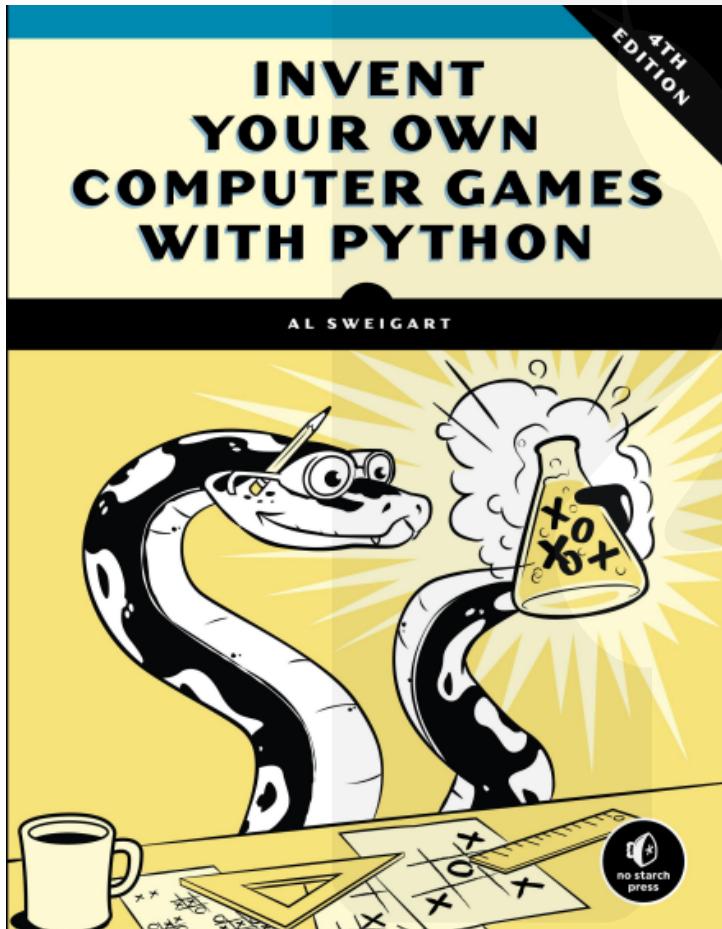
دانشگاه علوم ریاضی

دانشگاه فردوسی مشهد



آشنایی مفرحانه با مقدمات برنامه نویسی در پایتون

طرح و نظارت: دکتر رضا قنبری



بازی سازی

pygame بخش اول:



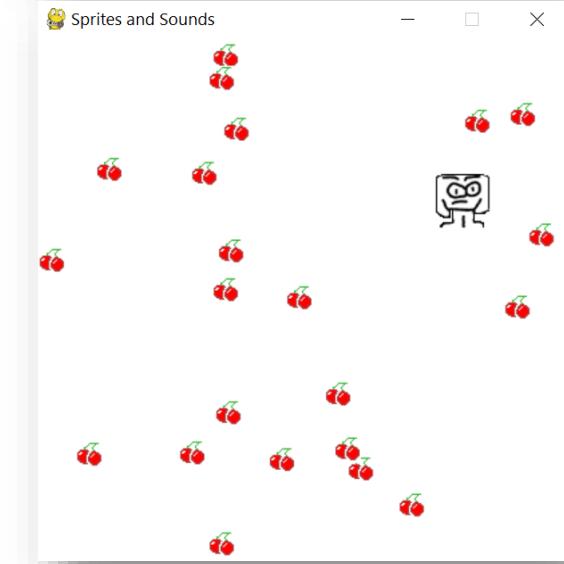
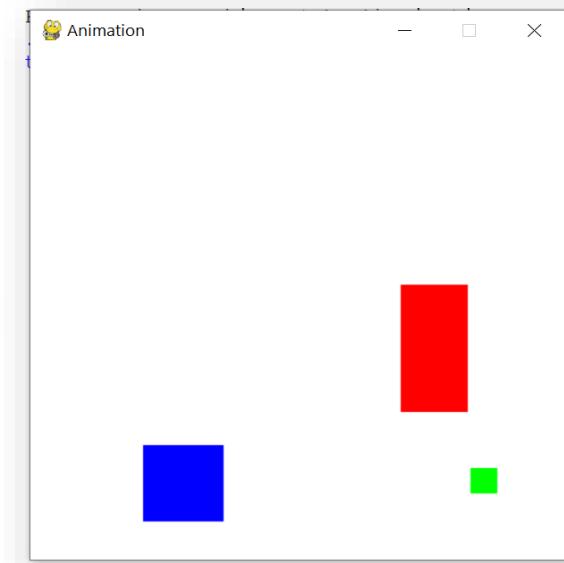
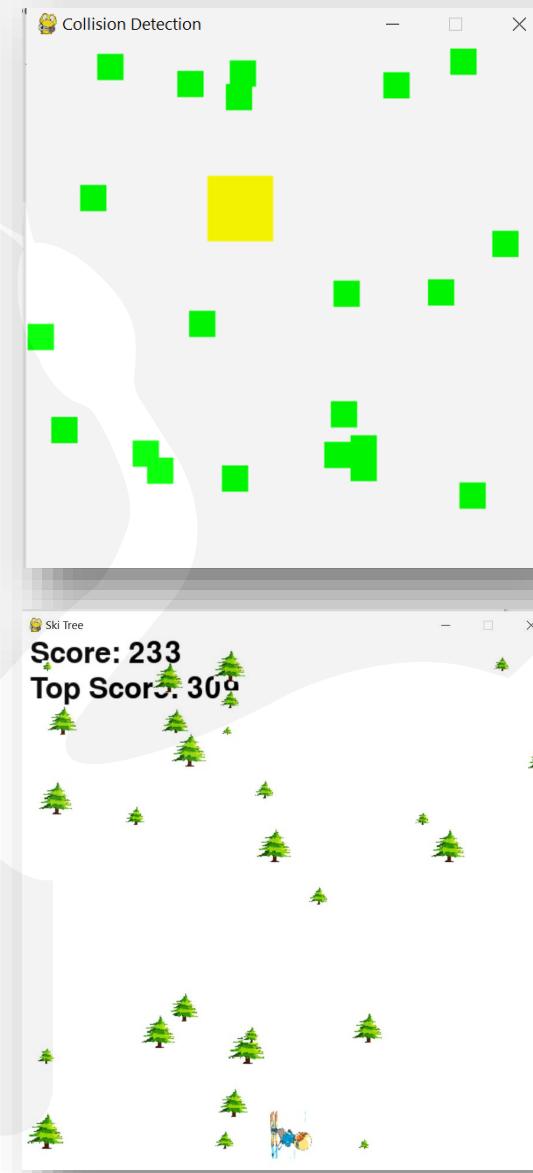
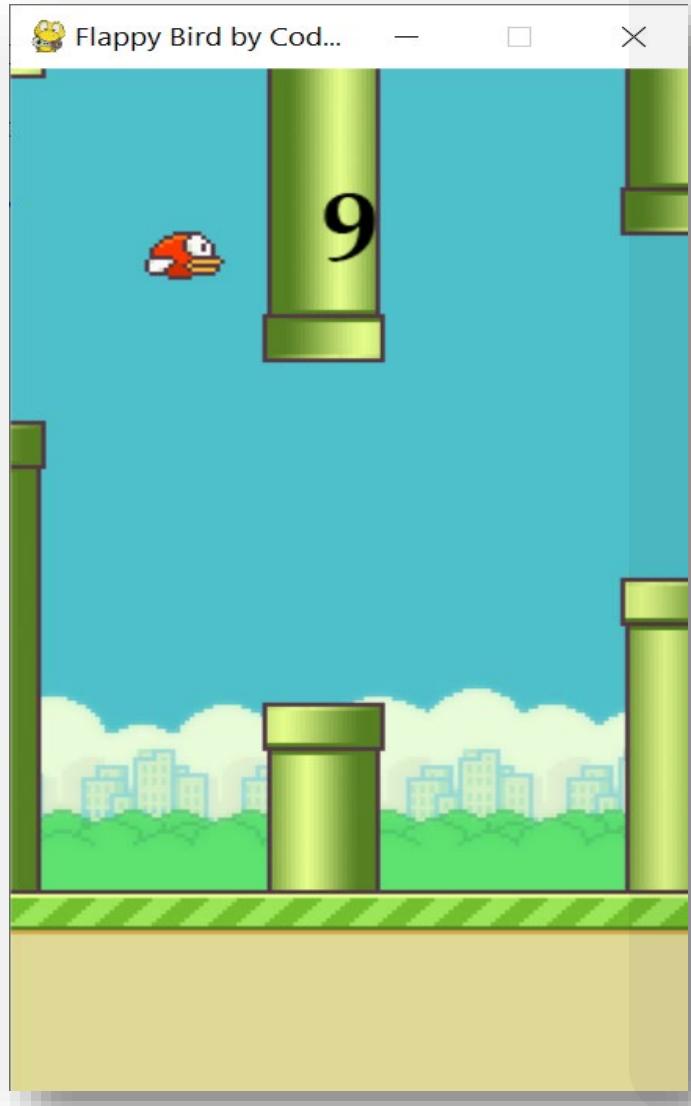
در این ترم ما می‌گیریم با استفاده از pygame برنامه‌های هیجان انگیز با گرافیک پیشرفته بسازیم. بازی‌های با گرافیک، انیمیشن و صدا. بعلاوه با استفاده از ماوس و صفحه کلید می‌توانید روی بازی خود کنترل داشته باشیم.

ماژول pygame به ما کمک می‌کند تا با راحت‌تر ترسیم کردن بازی روی سیستم و اضافه کردن موسیقی به برنامه، بازی بسازیم.

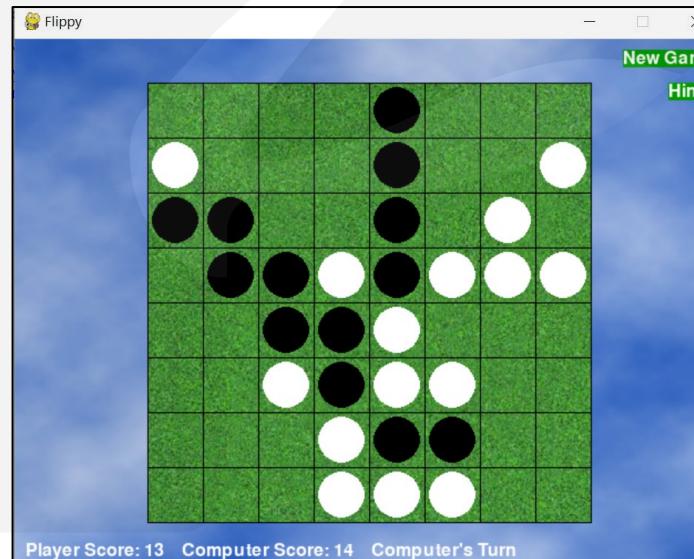
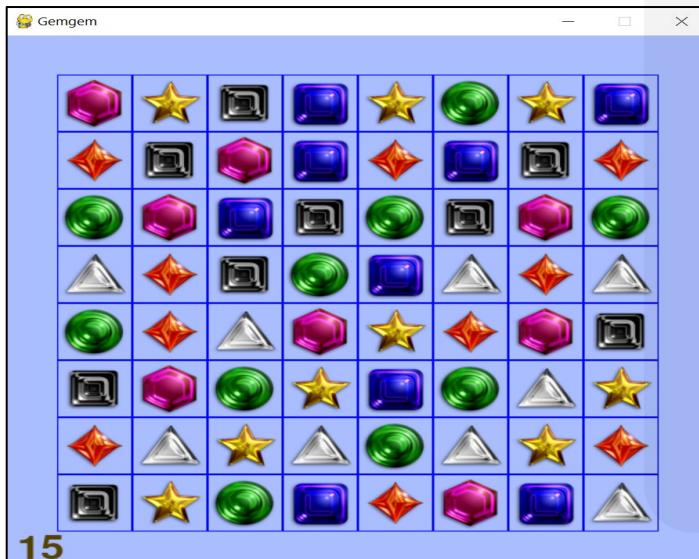
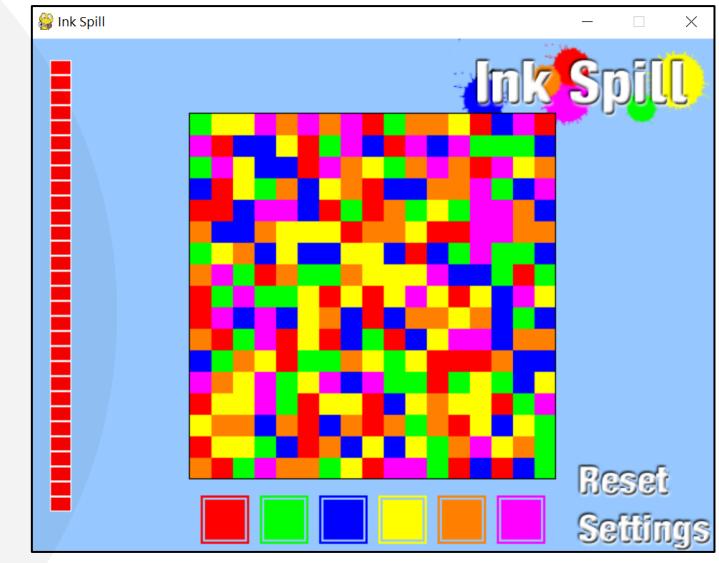
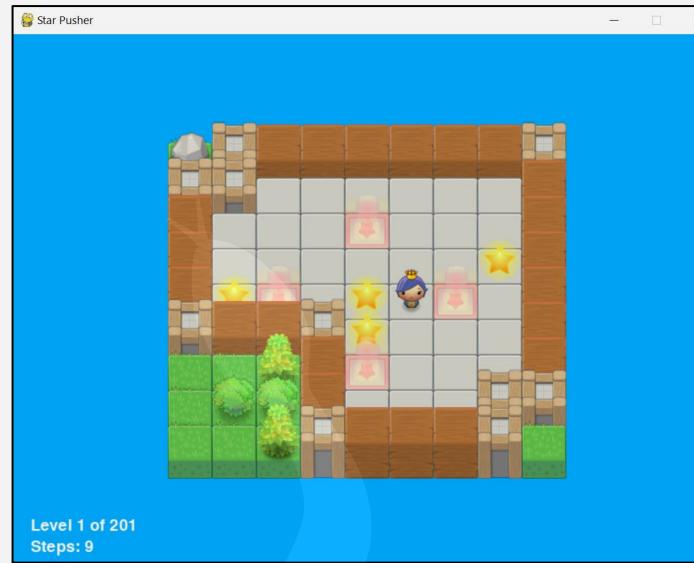
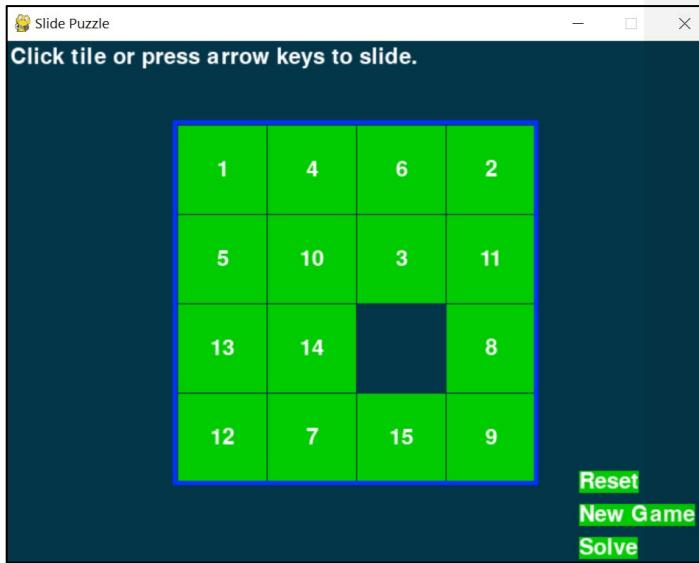


آمده است تا خلاقیت ما را به نمایش بگذارد! Pygame

بعضی از بازیهای که در این دوره می سازیم:



بعضی از بازیهای که میتوانیم با pygame بسازیم:





pygame

مشروع بايثون

HELLO WORLD!

در اولین قدم ما میخواهیم hello world را در pygame اجرا کنیم.

ماژول pygame با شل به درستی کار نمی کند و بنابراین شما باید برنامه را به در یک فایل بنویسید. شما نمی توانید دستورالعمل ها را یکی یکی در شل ارسال کنید. همچنین pygame از توابع input() و print() استفاده نمی کند. ورودی و خروجی متن وجود ندارد. در عوض pygame خروجی را با ترسیم گرافیک و متن در یک صفحه جدا نشان می دهد. ورودی pygame از طریق صفحه کلید و ماوس است.

نکته خوب استفاده از پنجره این است که متن شما می تواند در هر نقطه از پنجره ظاهر شود، نه فقط بعد از متن قبلی که شما چاپ کرده اید. متن همچنین می تواند در هر رنگ و اندازه ای باشد. پنجره مانند یک بوم است و شما می توانید هر چیزی را که دوست دارید روی آن بکشید.

وارد کردن مازول pygame



باید شروع کنیم.

ابتدا باید مازول pygame را وارد می کنیم تا بتوانیم توابع آن را فراخوانی کنیم. شما می توانید با گذاشتن کاما بین نام مازولها، چند مازول را در یک خط وارد کنید:

```
import pygame, sys
```

حالا مازول pygame.locals را وارد می کنیم:

```
from pygame.locals import *
```

این مازول شامل بسیاری از متغیرهای ثابت است که شما با pygame استفاده خواهید کرد، مانند QUIT، که به شما کمک می کند تا از برنامه خارج. کد بالا همچنین به شما امکان می دهد بدون نیاز به تایپ locals در مقابل هر متده استفاده کنید، یا هر چیز دیگری که از مازول فراخوانی می کنید، از مازول pygame.locals استفاده کنید.

راه اندازی pygame

همه برنامه های pygame را پس از وارد کردن ماژول pygame.init () باید pygame فراخوانی سایر توابع pygame کند:

```
pygame.init()
```

این عبارت، pygame را مقداردهی اولیه می کند، بنابراین آماده استفاده است. شما نیازی به دانستن اینکه init چه کاری انجام می دهد نیستید. فقط باید به یاد داشته باشید که قبل از استفاده از توابع دیگر pygame آن را فراخوانی کنید.



راه اندازی پنجره pygame

با فراخوانی متد () set_mode() در مژول pygame.display یک پنجره رابط کاربری گرافیکی (GUI) ایجاد می شود.

```
pygame.display.set_caption('Hello world!')
```

```
windowSurface = pygame.display.set_mode((500, 400), 0, 32)
```

این متدها به تنظیم پنجره ای برای اجرای pygame کمک می کند.

با استفاده از تاپل یک پنجره با عرض ۵۰۰ پیکسل و ارتفاع ۴۰۰ پیکسل ایجاد می کنیم. پارامترهای دوم و سوم گزینه های پیشرفته ای فراتر از این دوره هستند. فقط برای آنها به ترتیب ۰ و ۳۲ را وارد کنید.

متد () set_caption() فقط عنوان پنجره را به صورت Hello World! تنظیم می کند. عنوان در سمت چپ بالای پنجره است.



تنظیم متغیرهای رنگ

سه رنگ اصلی نور برای پیکسل ها وجود دارد: قرمز، سبز و آبی. با ترکیب مقدارهای مختلف این سه رنگ می‌توانید هر رنگ دیگری را ایجاد کنید. در pygame رنگ ها با سه عدد صحیح نشان داده می‌شوند. برای آن که هر بار که می‌خواهیم از یک رنگ خاص استفاده کنیم، یک تاپل سه عددی را بازنویسی نکنیم، می‌توانیم ثابت هایی را برای نگه داشتن تاپل بسازیم که رنگ تاپل را نشان دهد.

مثل زیر:

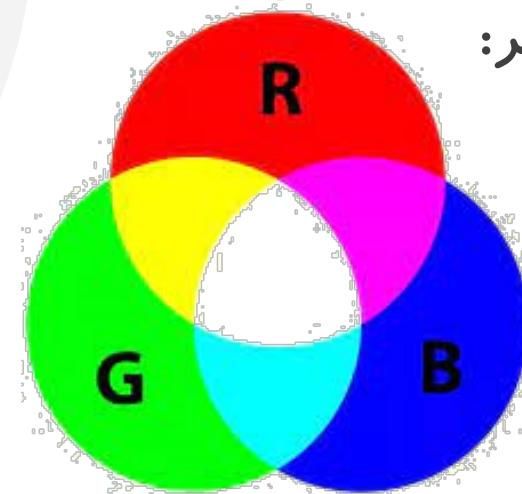
BLACK = (0, 0, 0)

WHITE = (255, 255, 255)

RED = (255, 0, 0)

GREEN = (0, 255, 0)

BLUE = (0, 0, 255)



اولین مقدار در تاپل تعیین می‌کند که چقدر قرمز در رنگ است. مقدار دوم مربوط به رنگ سبز و مقدار سوم برای آبی است. این سه عدد صحیح یک تاپل RGB را تشکیل می‌دهند

نوشتن متن در پنجره pygame

نوشتن متن روی یک پنجره با استفاده از `print()` در بازی های مبتنی بر متن کمی متفاوت است.
برای نوشتن متن روی یک پنجره، به تنظیمات اولیه نیاز داریم.



اولین مورد فونت نوشته است. نوشتن متن روی یک پنجره با استفاده از `print()` در بازی های مبتنی بر متن کمی متفاوت است. شکل مقابل یک جمله را نشان می دهد که با فونت های مختلف چاپ شده است.

نوشتن متن در پنجره pygame



در بازی های مبتنی بر متن ما فقط به پایتون می گفتیم که متن را چاپ کند. رنگ، اندازه، و فونتی که برای نمایش این متن استفاده شده بود کاملاً توسط سیستم عامل شما تعیین می شد و برنامه پایتون نمی توانست فونت را تغییر دهد. اما pygame می تواند متن را با هر قلمی در رایانه شما ترسیم کند.

کد زیر با فراخوانی تابع `pygame.font.SysFont()` با دو پارامتر، یک شیء ایجاد می کند :

```
basicFont = pygame.font.SysFont(None, 48)
```

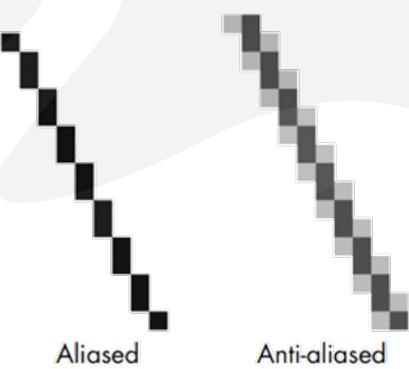
اولین پارامتر نام فونت است. مقدار `None` از فونت پیش فرض سیستم استفاده می کند. پارامتر دوم اندازه فونت است..

رender کردن یک فونت

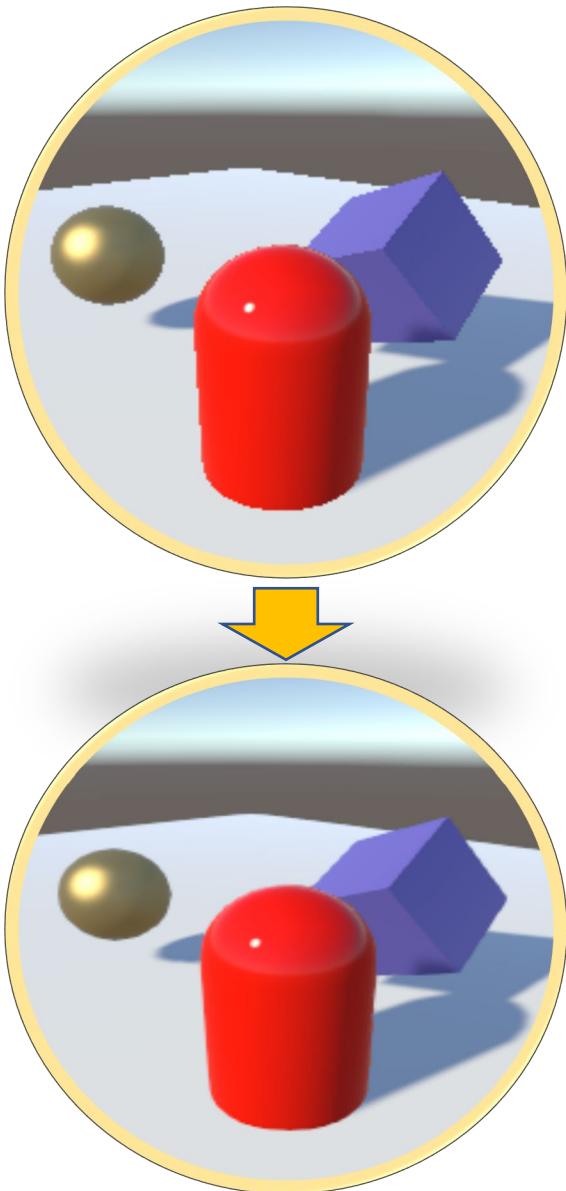
حالا که فونت را مشخص کردیم باید تصویر متن را ایجاد کنیم. به این کار render کردن می گویند. یک متده BasicFont render() دارد که یک سطح را با متنی که بر روی آن نوشته شده ایجاد می کند.

```
text = basicFont.render('Hello world!', True, WHITE, BLUE)
```

اولین پارامتر render() متن مورد نظرمان است. پارامتر دوم یک Boolean برای anti-aliasing کردن یا نکردن فونت است. anti-aliasing متن شما را کمی تار می کند تا صاف تر به نظر برسد. شکل زیر نشان می دهد که یک خط با و بدون anti-aliasing چگونه به نظر می رسد.



در کد بالا true نشان می دهد که از anti-aliasing استفاده کرده ایم. پارامتر سوم رنگ متن است و پارامتر چهارم رنگ پس زمینه متن است.



پر کردن سطح با رنگ

تابع `fill()` سطح را به طور کامل با رنگی که به عنوان پارامتر وارد می کنید پرمی کند.

```
windowSurface.fill((255, 255, 255))
```

توجه داشته باشید که در pygame ، وقتی متده `fill()` یا هر یک از توابع ترسیمی دیگر را فراخوانی کنید پنجره روی صفحه شما تغییر نمی کند. بلکه، اینها شیء Surface را تغییر می دهند.



قرار دادن پس زمینه



اگر بخواهیم در پنجره پس زمینه ایجاد کنیم باید اول آنرا بارگذاری کنیم. سپس به صفحه منتقل کنیم.

```
bg = pygame.image.load("bg.png")
windowSurface.blit(bg, (0, 0))
```

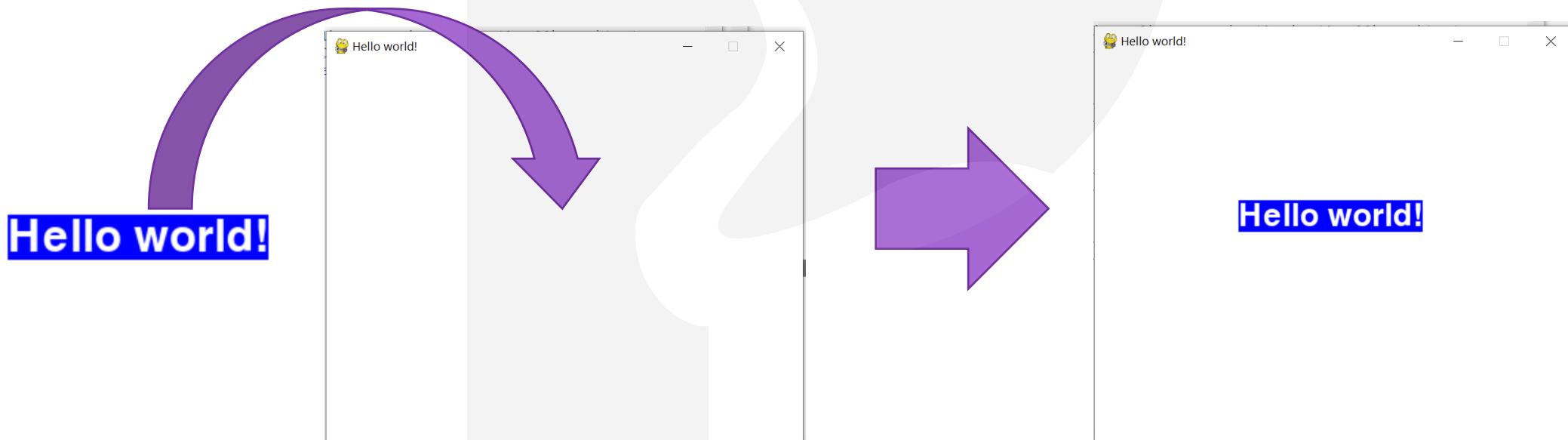
فقط باید حواسمن باشد که عکس را در مسیر اجرای فایل قرار دهیم.

تنظیم مکان متن با ویژگی های Rect

برای آن که تصویر متن خود را به صفحه برنامه منتقل کنیم از متده استفاده می کنیم:

```
windowSurface.blit(text, (150, 150) )
```

در کد بالا اولین پارامتر، تصویر رندر شده است و دومین پارامتر تاپل شامل مختصات x و y مکانی است که متن در صفحه قرار می گیرد.



ترسیم شیء سطح روی صفحه

در تا زمانی که تابع `pygame.display.update()` فراخوانی نشده، هیچ چیز در صفحه نمایش کشیده نمی شود، پس ما آن را فراخوانی می کنیم تا سطح به روز رسانی شده خود را نمایش دهیم.

`pygame.display.update()`

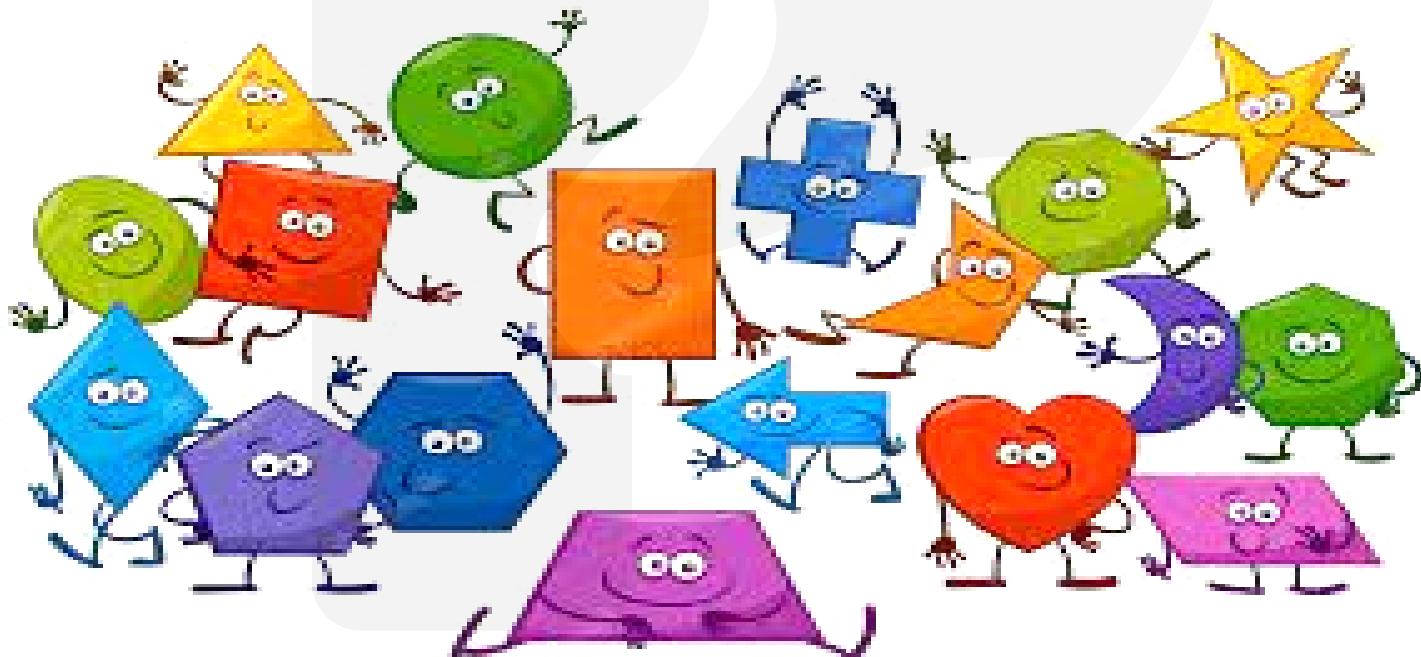
برای صرفه جویی در حافظه صفحه نمایش را فقط یک بار پس از فراخوانی تمام توابع ترسیمی به روز کنید.



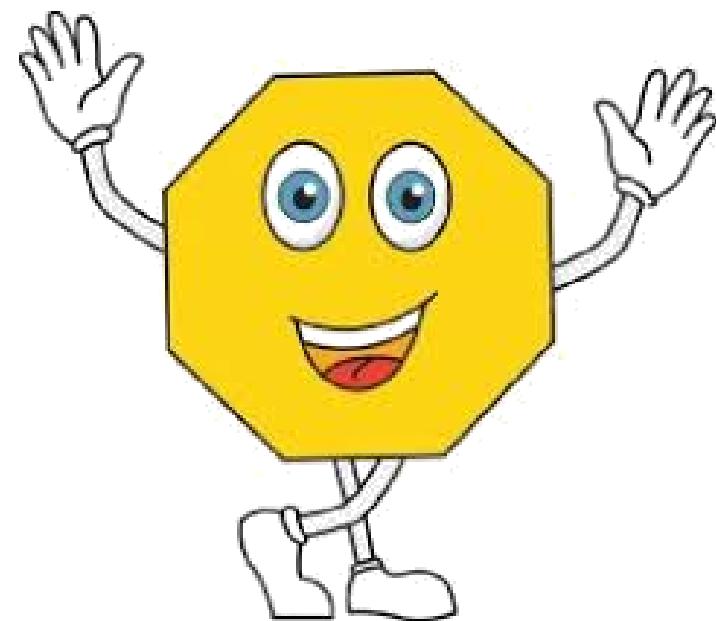
توابع ترسیم pygame

تا اینجا، ما یاد گرفتیم که چگونه یک پنجره pygame را با یک رنگ پر کنیم و متن اضافه کنیم و با استفاده از تابع `update()` توانستیم متن خود را نمایش دهیم.

علاوه بر این دارای توابعی است که به شما امکان می دهد اشکال و خطوط را ترسیم کنید. هر شکل عملکرد خاص خود را دارد و می توانید این اشکال را در تصاویر مختلف برای بازیهای گرافیکی خود ترکیب کنید.



رسم چند ضلعی

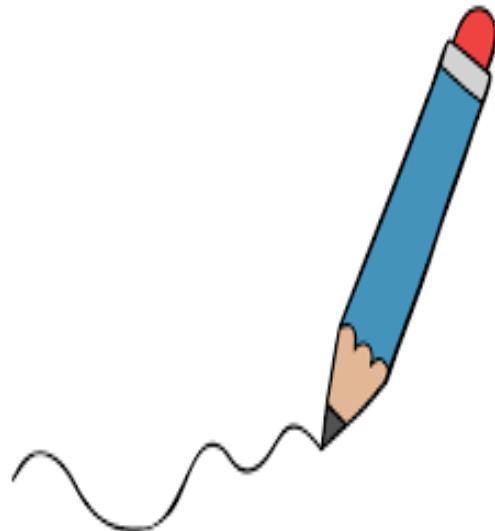


تابع `pygame.draw.polygon()` می تواند هر شکل چند ضلعی را که به آن می دهید ترسیم کند. آرگومان های این تابع به ترتیب : سطحی که چند ضلعی در آن رسم می شود، رنگ چند ضلعی، تاپلی که مختصات `x` و `y` نقاط را نشان می دهد (برای تکمیل شکل آخرین تاپل به طور خودکار به اولی متصل می شود)، در صورت تمایل، یک عدد صحیح برای عرض خطوط چند ضلعی(در صورت صفر بودن این مورد، چند ضلعی توپر می شود).

```
pygame.draw.polygon(windowSurface, (0,255,0), ((146, 0), (291,  
106), (236, 277), (56, 277), (0, 106)), 4)
```

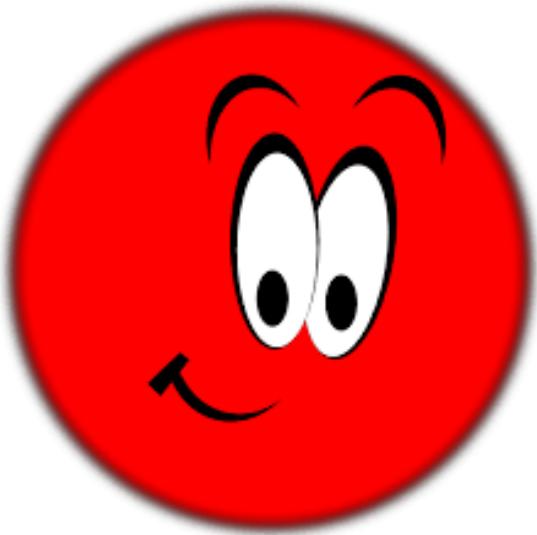
کشیدن یک خط

تابع `pygame.draw.line()` فقط یک خط از یک نقطه روی صفحه نمایش به نقطه دیگر رسم می کند. آرگومان های این تابع به ترتیب : سطحی که خط در آن رسم می شود، رنگ خط، تاپلی از دو عدد صحیح برای مختصات `x` و `y` انتهای دیگر خط، در صورت تمایل، یک عدد صحیح برای ضخامت خط



```
pygame.draw.line(windowSurface, (0, 0, 255), (60, 60), (120, 60), 4)
```

رسم دایره

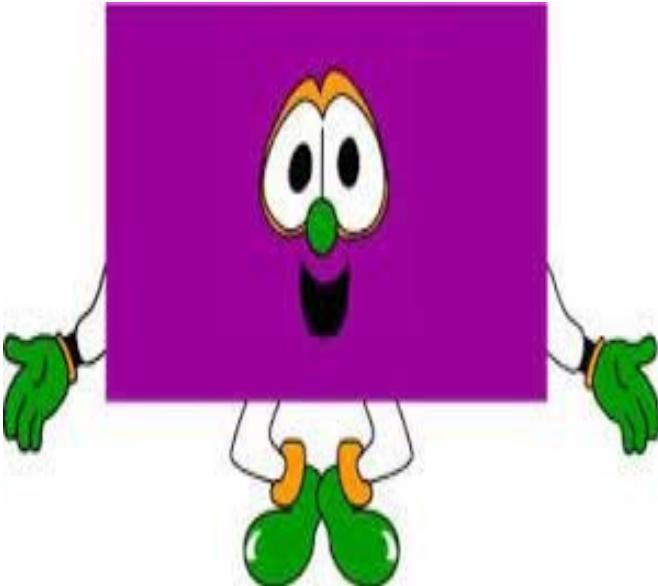


تابع `pygame.draw.circle()` Surface را روی دایره هایی را می ترسیم کند. آرگومان های این تابع به ترتیب : سطحی که دایره در آن رسم می شود، رنگ دایره، تاپلی از دو عدد صحیح برای مختصات x و y مرکز دایره، یک عدد صحیح برای شعاع دایره، در صورت تمایل، یک عدد صحیح برای عرض خط (عرض ۰ به این معنی است که دایره توپر خواهد شد).

```
pygame.draw.circle(windowSurface, (255, 0, 0), (300, 50), 20, 0)
```

رسم مستطیل

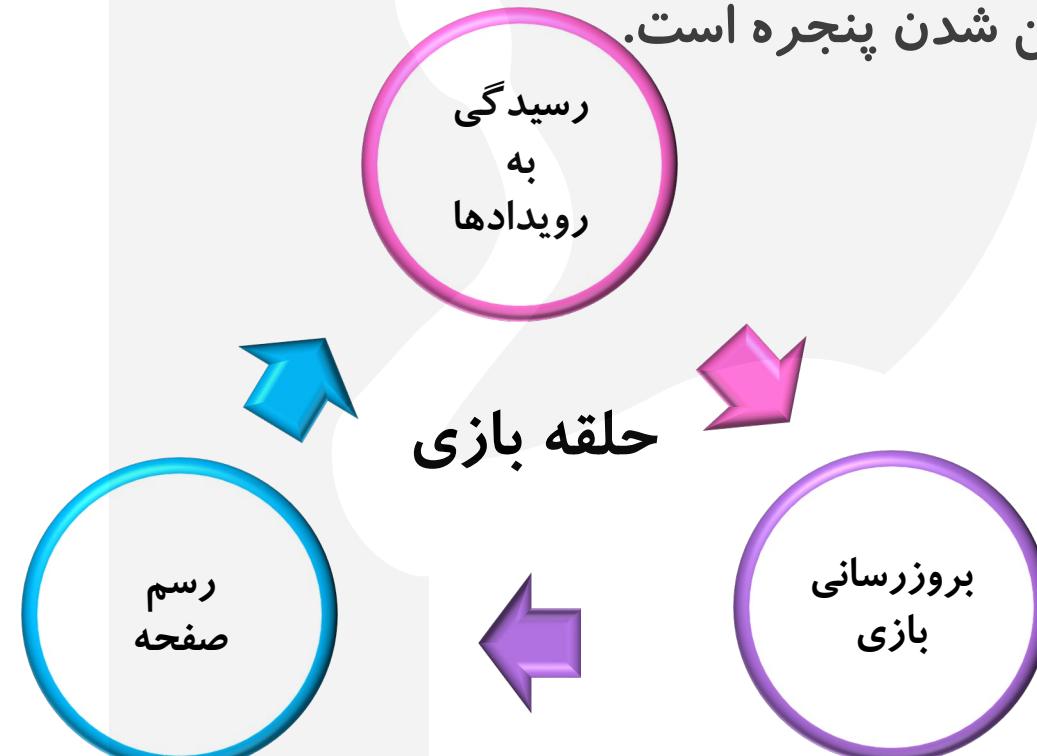
تابع `pygame.draw.rect()` یک مستطیل رسم می کند. آرگومان های این تابع به ترتیب : سطحی که مستطیل در آن رسم می شود، رنگ مستطیل، تاپلی از چهار عدد صحیح برای مختصات `x` و `y` گوشه بالا سمت چپ و عرض و ارتفاع مستطیل، در صورت تمایل، یک عدد صحیح برای عرض خط (عرض ۰ به این معنی است که مستطیل توپر خواهد شد).



```
pygame.draw.rect(windowSurface, (0,0,255), (10, 10, 40, 80), 1)
```

حلقه بازی

برنامه های pygame به طور مداوم از طریق یک حلقه بازی در حال اجرا هستند. این حلقه هرگز متوقف نمی شود مگر این که شرط آن False شود. به این حلقه حلقه اصلی بازی یا main loop می گوییم. کار آن اجرای بازی تا هنگام بستن شدن پنجره است.



حلقه بازی

حلقه بازی به این صورت است:

```
while True:  
    for event in pygame.event.get():  
        if event.type == QUIT:  
            pygame.quit()  
            sys.exit()
```

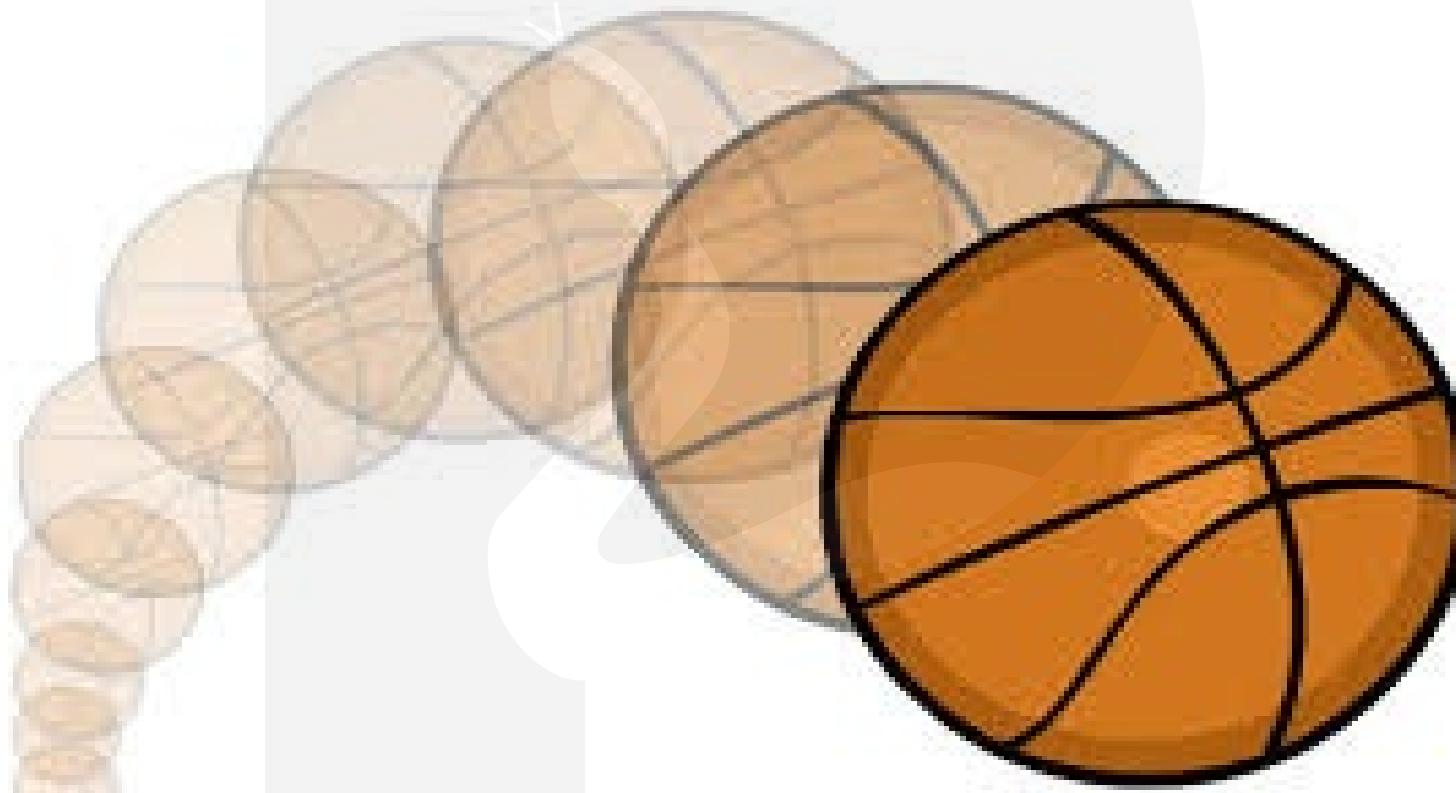


حلقه بازی

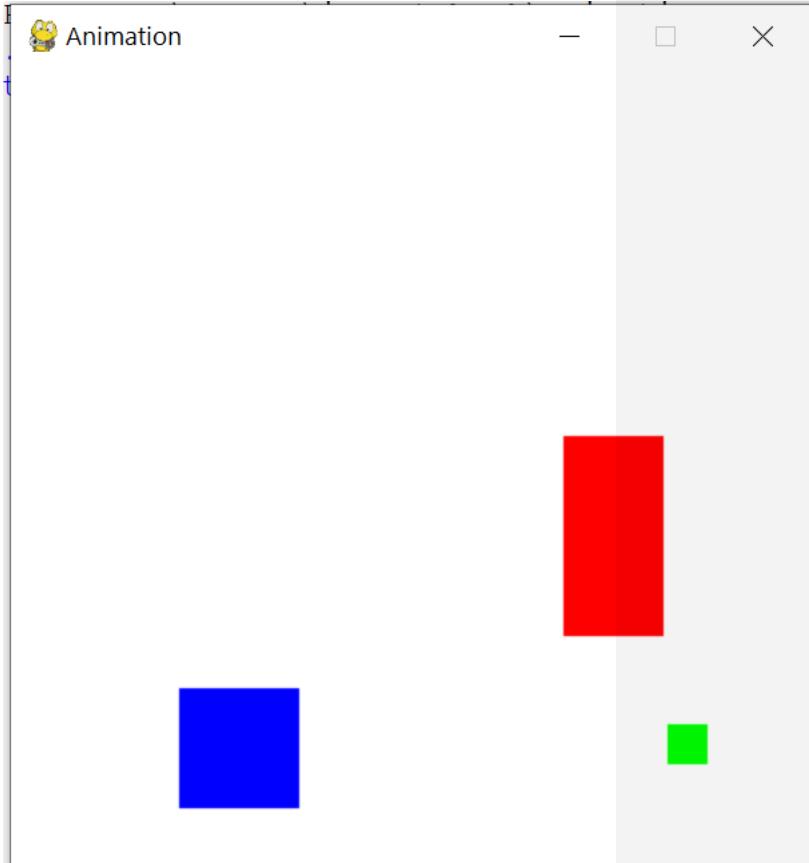
هر بار که کاربر کاری را انجام می دهد مانند فشار دادن صفحه کلید یا حرکت موس روی پنجره برنامه، یک شیء pygame.event.Event ایجاد می شود تا این رویدادها را ثبت کند. با فراخوانی تابع pygame.event.get () می توانیم بفهمیم که کدام رویدادها رخ داده اند.

اشیاء event یک ویژگی به نام type دارند که مشخص کننده نوع رویداد است. کد بررسی می کند که آیا نوع رویداد برابر با مقدار ثابت QUIT است یا نه. اگر یک رویداد quit داشته باشیم، آنگاه توابع pygame.quit و sys.exit فراخوانی می شوند. تابع pygame.quit این تابع کدی را اجرا می کند که کتابخانه pygame را غیرفعال می کند.

انیمیشن ,,pygame



انیمیشن



اکنون که چند مهارت pygame را یاد گرفتید، برنامه ای برای متحرک سازی جعبه هایی می نویسیم که دور یک پنجره می پرند. جعبه ها رنگ ها و اندازه های متفاوتی دارند و فقط در جهت های مورب حرکت می کنند. برای متحرک سازی جعبه ها، آنها را در هر تکرار در حلقه بازی چند پیکسل حرکت می دهیم. این کار شبیه این به نظر می رسد که جعبه ها در حال حرکت در اطراف صفحه نمایش هستند.

کد اینیمیشن

```
import pygame, sys, time
from pygame.locals import *

# Set up pygame.
pygame.init()

# Set up the window.
WINDOWWIDTH = 400
WINDOWHEIGHT = 400
windowSurface = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT), 0, 32)
pygame.display.set_caption('Animation')

# Set up direction variables.
DOWNLEFT = 'downleft'
DOWNRIGHT = 'downright'
UPLEFT = 'upleft'
UPRIGHT = 'upright'

MOVESPEED = 4
```

کد اینیمیشن

```
# Set up the colors.  
WHITE = (255, 255, 255)  
RED = (255, 0, 0)  
GREEN = (0, 255, 0)  
BLUE = (0, 0, 255)  
  
# Set up the box data structure.  
b1 = {'rect':pygame.Rect(300, 80, 50, 100), 'color':RED, 'dir':UPRIGHT}  
b2 = {'rect':pygame.Rect(200, 200, 20, 20), 'color':GREEN, 'dir':UPLEFT}  
b3 = {'rect':pygame.Rect(100, 150, 60, 60), 'color':BLUE, 'dir':DOWNLEFT}  
boxes = [b1, b2, b3]  
  
# Run the game loop.  
while True:  
    # Check for the QUIT event.  
    for event in pygame.event.get():  
        if event.type == QUIT:  
            pygame.quit()  
            sys.exit()
```

کد انیمیشن

```
# Draw the white background onto the surface.  
windowSurface.fill(WHITE)  
  
for b in boxes:  
    # Move the box data structure.  
    if b['dir'] == DOWNLEFT:  
        b['rect'].left -= MOVESPEED  
        b['rect'].top += MOVESPEED  
    if b['dir'] == DOWNRIGHT:  
        b['rect'].left += MOVESPEED  
        b['rect'].top += MOVESPEED  
    if b['dir'] == UPLEFT:  
        b['rect'].left -= MOVESPEED  
        b['rect'].top -= MOVESPEED  
    if b['dir'] == UPRIGHT:  
        b['rect'].left += MOVESPEED  
        b['rect'].top -= MOVESPEED
```

کد انیمیشن

```
# Check whether the box has moved out of the window.  
if b['rect'].top < 0:  
    # The box has moved past the top.  
    if b['dir'] == UPLEFT:  
        b['dir'] = DOWNLEFT  
    if b['dir'] == UPRIGHT:  
        b['dir'] = DOWNRIGHT  
if b['rect'].bottom > WINDOWHEIGHT:  
    # The box has moved past the bottom.  
    if b['dir'] == DOWNLEFT:  
        b['dir'] = UPLEFT  
    if b['dir'] == DOWNRIGHT:  
        b['dir'] = UPRIGHT  
if b['rect'].left < 0:  
    # The box has moved past the left side.  
    if b['dir'] == DOWNLEFT:  
        b['dir'] = DOWNRIGHT  
    if b['dir'] == UPLEFT:  
        b['dir'] = UPRIGHT
```

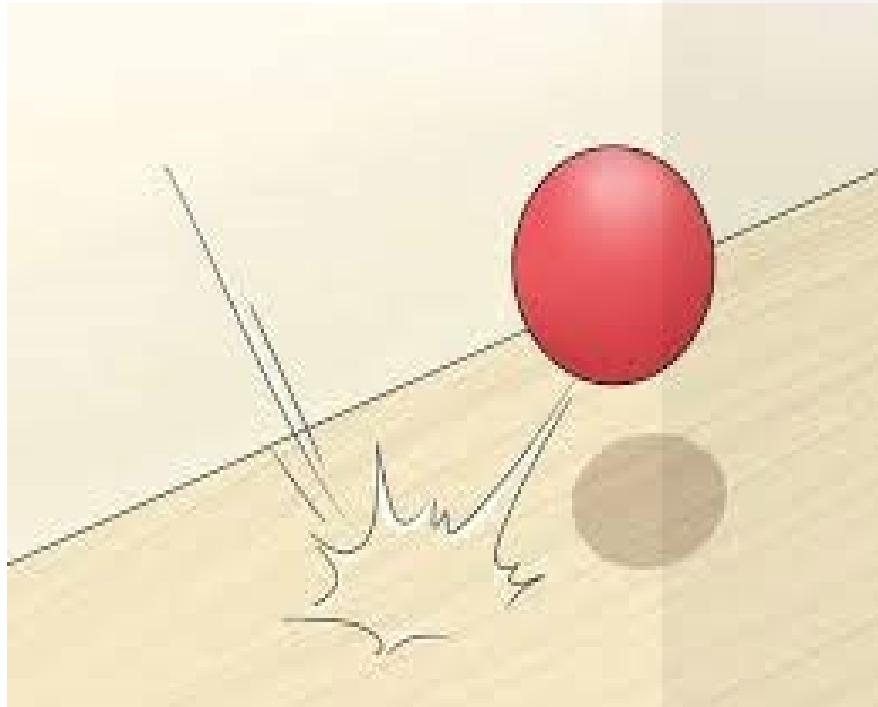
کد اینیمیشن

```
if b['rect'].right > WINDOWWIDTH:  
    # The box has moved past the right side.  
    if b['dir'] == DOWNRIGHT:  
        b['dir'] = DOWNLEFT  
    if b['dir'] == UPRIGHT:  
        b['dir'] = UPLEFT  
  
    # Draw the box onto the surface.  
    pygame.draw.rect(windowSurface, b['color'], b['rect'])  
  
# Draw the window onto the screen.  
pygame.display.update()  
time.sleep(0.02)
```

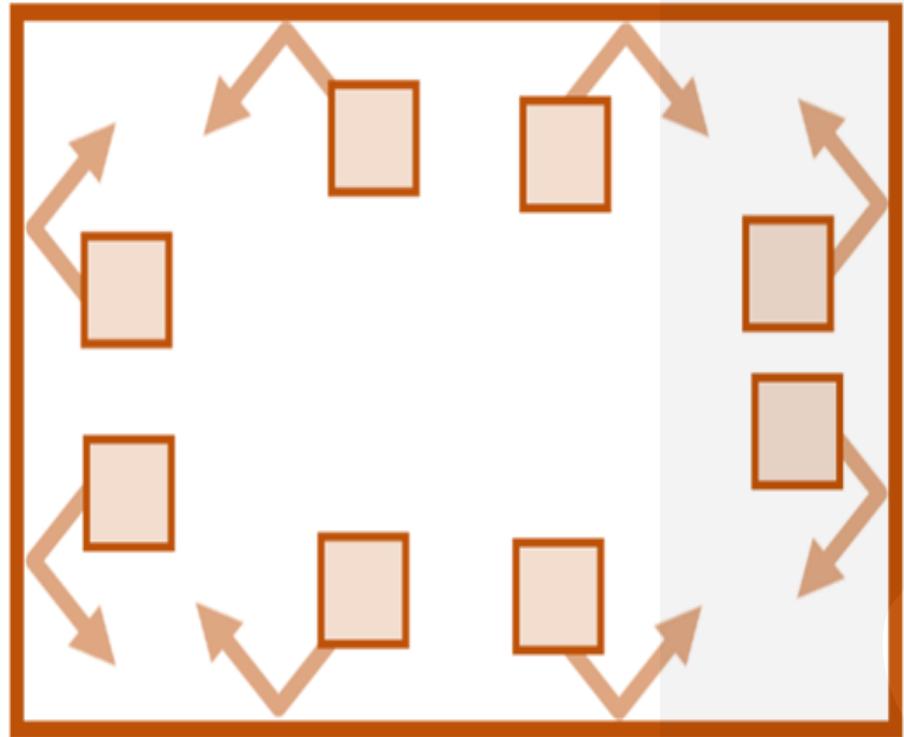
حرکت و پرش جعبه ها

در این برنامه سه جعبه با رنگ های مختلف داریم که در حال حرکت و پریدن از دیوارهای پنجره هستند.

ابتدا باید به این فکر کنیم که جعبه ها چگونه حرکت کنند. هر جعبه در یکی از چهار جهت مورب حرکت می کند. وقتی یک جعبه به کنار پنجره برخورد می کند، باید به صورت مورب در جهت جدید حرکت کند.



حرکت و پرش جعبه ها



جهت جدید حرکت جعبه پس از برخورد با دیوار به دو مورد بستگی دارد: قبل از برخورد در چه جهتی حرکت می کرده و به کدام دیوار برخورد کرده است. هشت راه ممکن برای یک جعبه وجود دارد: دو روش متفاوت برای هر یک از چهار دیوار. به عنوان مثال، اگر یک جعبه در حال حرکت به سمت پایین و سمت راست است و سپس به لبه پایینی پنجره برخورد می کند، جهت جدید آن بالا و راست است.

شروع نوشتن کد

ابتدا باید مشابه قسمت قبل ماثولهای لازم را وارد و پس از آن pygame را مقداردهی اولیه کنیم. سپس پنجره با عرض و ارتفاع مشخص را ایجاد و برای آن نام

```
import pygame, sys, time
from pygame.locals import *

pygame.init()

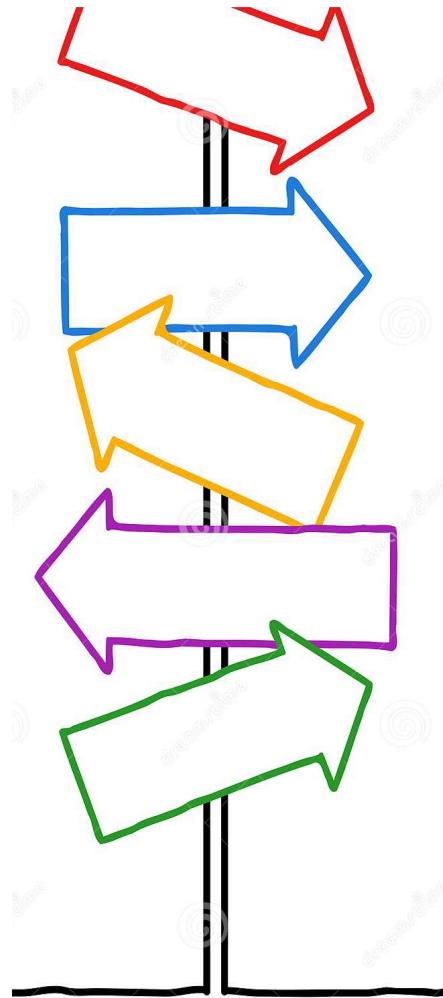
WINDOWWIDTH = 400
WINDOWHEIGHT = 400
windowSurface = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT), 0, 32)
pygame.display.set_caption('Animation')
```

انتخاب می کنیم:



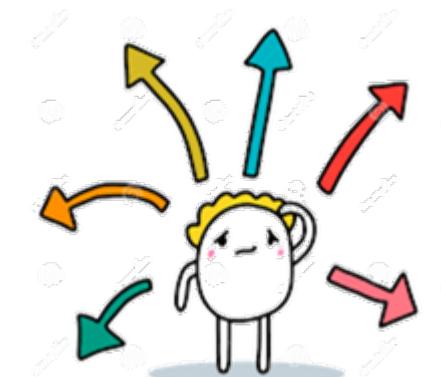
چون عرض و ارتفاع پنجره در طول اجرای برنامه هرگز تغییر نمی کنند، متغیرهای ثابت برای ابعاد پنجره ایده خوبی هستند.

تنظیم متغیرهای ثابت



از متغیرهای ثابت همچنین برای هر یک از چهار جهتی که جعبه ها می توانند حرکت کنند استفاده می کنیم:

```
DOWNLEFT = 'downleft'  
DOWNRIGHT = 'downright'  
UPLEFT = 'upleft'  
UPRIGHT = 'upright'
```



همچنین یک متغیر ثابت برای تعیین سرعت جعبه ها ایجاد می کنیم:

```
MOVESPEED = 4
```

مقدار ۴ در متغیر ثابت MOVESPEED به برنامه می گوید که جعبه در هر تکرار در حلقه بازی چه تعداد پیکسل باید حرکت کند.

متغیرهای ثابت برای رنگ

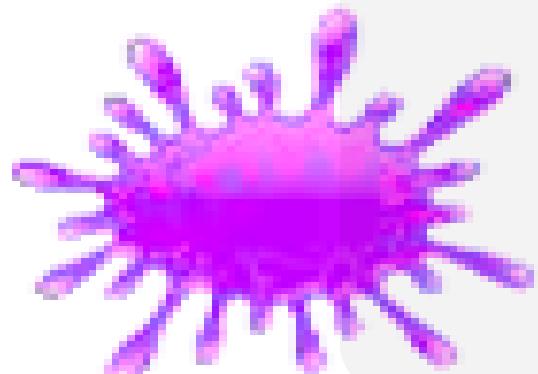
همانطور که گفتیم، pygame از سه مقدار صحیح برای مقادیر قرمز، سبز و آبی استفاده می کند که RGB نامیده می شود و بازه اعداد صحیح از ۰ تا ۲۵۵ است.

WHITE = (255, 255, 255)

$$\text{RED} = (255, 0, 0)$$

GREEN = (0, 255, 0)

$$\text{BLUE} = (0, 0, 255)$$



از متغیرهای ثابت برای خوانایی پیشتر برنامه استفاده می شود.

ایجاد جعبه ها

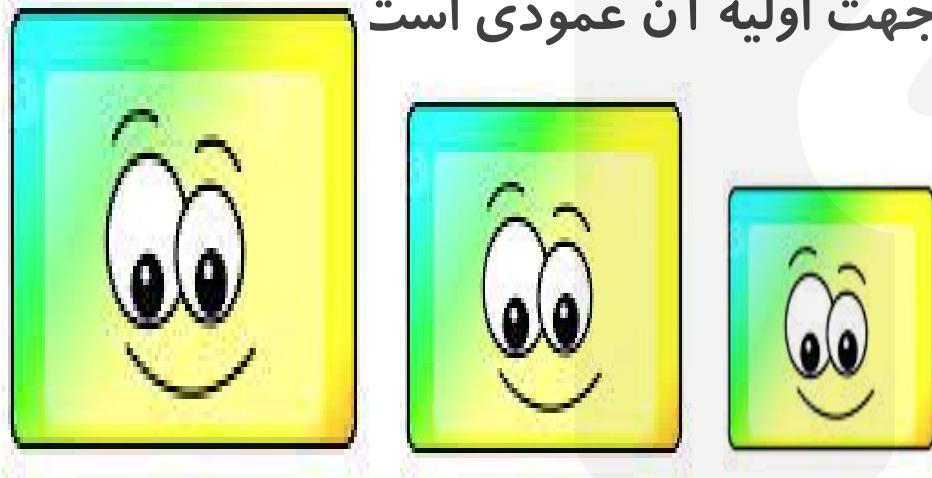
حالا می خواهیم جعبه ها را ایجاد کنیم. پس برای هر جعبه متحرک یک دیکشنری تعریف می کنیم.

متغیر b1 اولین جعبه ماست:

```
b1 = {'rect':pygame.Rect(300, 80, 50, 100), 'color':RED, 'dir':UPRIGHT}
```

گوش سمت چپ بالای این جعبه در مختصات $x = 300$ و $y = 80$ قرار دارد و عرض جعبه ۵۰

پیکسل و ارتفاع آن ۱۰۰ پیکسل است. رنگ آن قرمز و جهت اولیه آن عمودی است



ایجاد جعبه ها

b2 و b3 دو جعبه دیگر با اندازه ها، موقعیت ها، رنگ ها و جهت های مختلف هستند:

```
b2 = {'rect':pygame.Rect(200, 200, 20, 20), 'color':GREEN, 'dir':UPLEFT}
b3 = {'rect':pygame.Rect(100, 150, 60, 60), 'color':BLUE, 'dir':DOWNLEFT}
```

سه دیکشنری b1، b2 و b3 در یک لیست در متغیر boxes

ذخیره می شوند.

```
boxes = [b1, b2, b3]
```



حلقه بازی

حلقه بازی کار اینیمیشن سازی جعبه های متحرک را انجام می دهد. هر جعبه ۴ پیکسل در هر تصویر حرکت خواهد کرد. اکنون که کمی در مورد نحوه عملکرد حلقه بازی می دانیم، باید کدش را بنویسیم.

هنگامی که بازیکن با بستن پنجره خارج می شود، باید برنامه را متوقف کنیم. برنامه ما دائمًا بررسی کند که یک رویداد QUIT وجود داشته است یا نه:

`while True:`

```
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
```

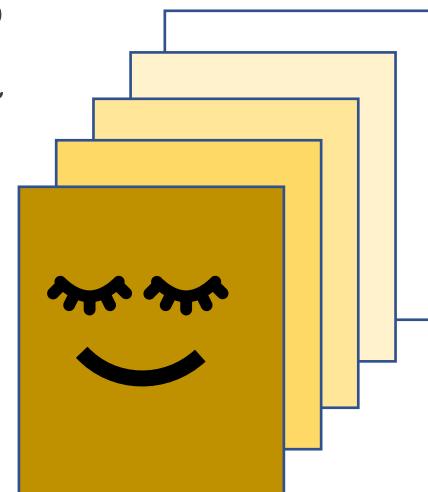


حلقه بازی

برای اینیمیشن دادن به جعبه ها در صفحه کد ما باید در هر بار تکرار حلقه، کل پنجره را با جعبه های جدیدی که هر بار چند پیکسل جلوتر قرار دارند، دوباره ترسیم کند.. اگر فقط به حلقه بازی اجازه دهیم جعبه ها روی صفحه نگه دارد، ما به جای اینیمیشن با دنباله ای از جعبه ها مواجه می شویم. برای جلوگیری از این وضعیت، باید پنجره را برای هر تکرار حلقه بازی پاک کنیم.

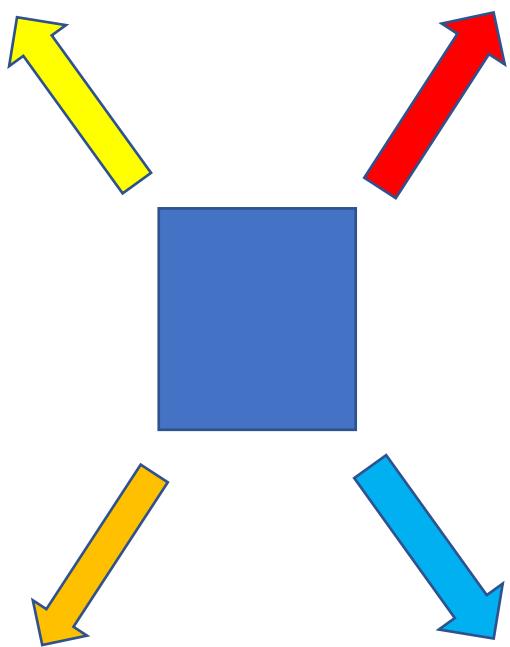
برای انجام این کار، کل سطح را با رنگ سفید پر می کنیم تا هر چیزی که قبل از روی آن کشیده شده است پاک شود:

```
windowSurface.fill(WHITE)
```



حرکت دادن هر جعبه

برای به روز رسانی موقعیت هر جعبه، باید داخل حلقه بازی روی لیست جعبه ها تکرار را انجام دهیم:

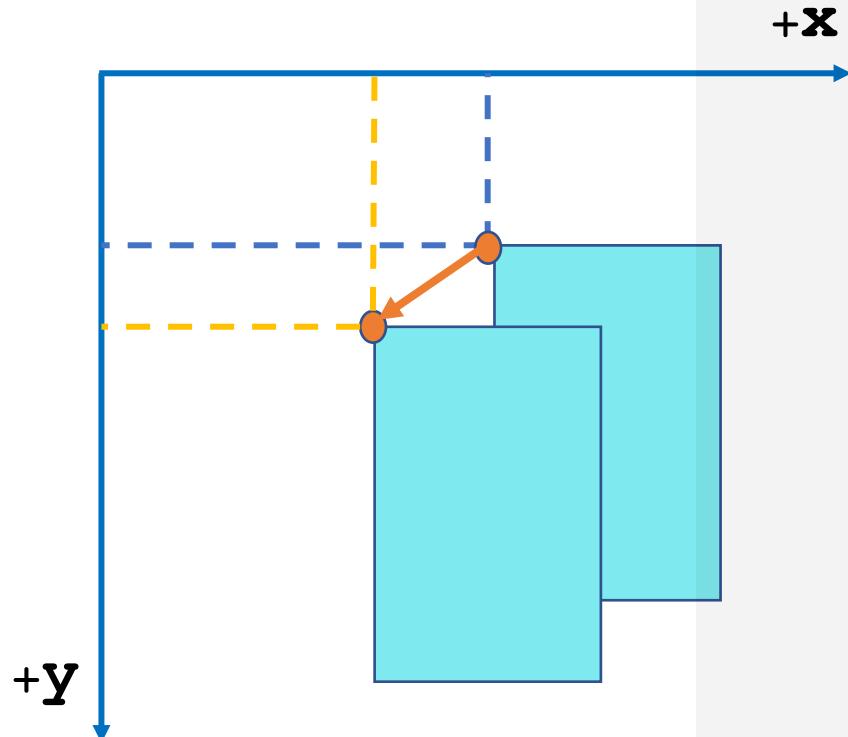


for b in boxes:

ما باید هر جعبه را بسته به جهتی که در حال حاضر در حال حرکت است، تغییر دهیم. بنابراین از دستورات if برای تعیین جهت جعبه استفاده خواهیم کرد. سپس بسته به جهتی که جعبه در حال حرکت است، موقعیت جعبه را تغییر می دهیم. مقدار پارامترهای left و top هر جعبه به میزان عدد صحیح ذخیره شده در MOVESPEED کم یا زیاد می شود، که تعداد پیکسل هایی است که جعبه در هر تکرار از طریق حلقه بازی حرکت می کند.

حرکت دادن هر جعبه

می دانیم که در `pygame.rect()` پارامترهای اول(x) و دوم(y) به ترتیب مختصات چپ و بالای جعبه هستند. حالا اگر حرکت در جهت پایین و چپ (DOWNLEFT) باشد، باید از مقدار x کم شود و به مقدار y اضافه گردد.

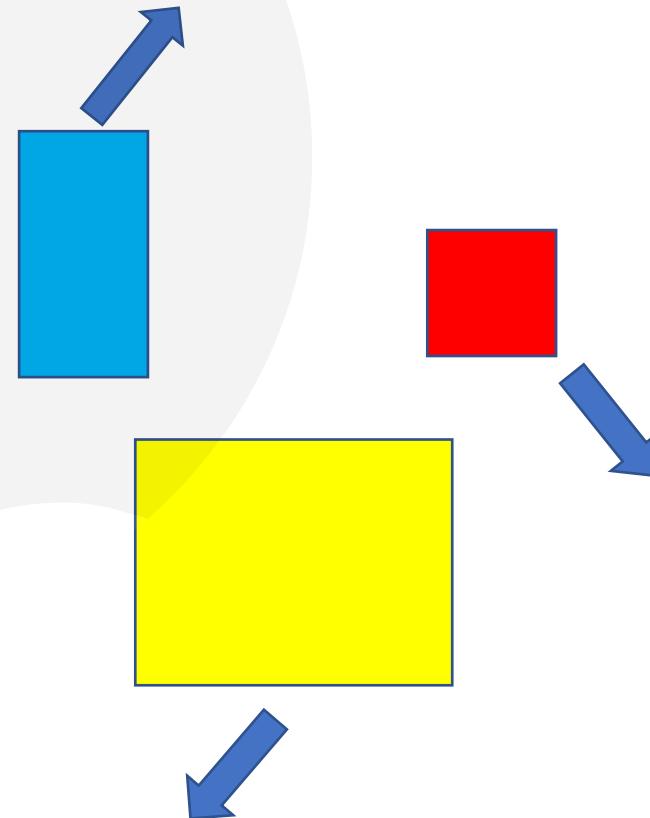


```
if b['dir'] == DOWNLEFT:  
    b['rect'].left -= MOVESPEED  
    b['rect'].top += MOVESPEED
```

حرکت دادن هر جعبه

به همین ترتیب این شرط برای همه جهت ها تعریف می شود.

```
if b['dir'] == DOWNLEFT:  
    b['rect'].left -= MOVESPEED  
    b['rect'].top += MOVESPEED  
if b['dir'] == DOWNRIGHT:  
    b['rect'].left += MOVESPEED  
    b['rect'].top += MOVESPEED  
if b['dir'] == UPLEFT:  
    b['rect'].left -= MOVESPEED  
    b['rect'].top -= MOVESPEED  
if b['dir'] == UPRIGHT:  
    b['rect'].left += MOVESPEED  
    b['rect'].top -= MOVESPEED
```

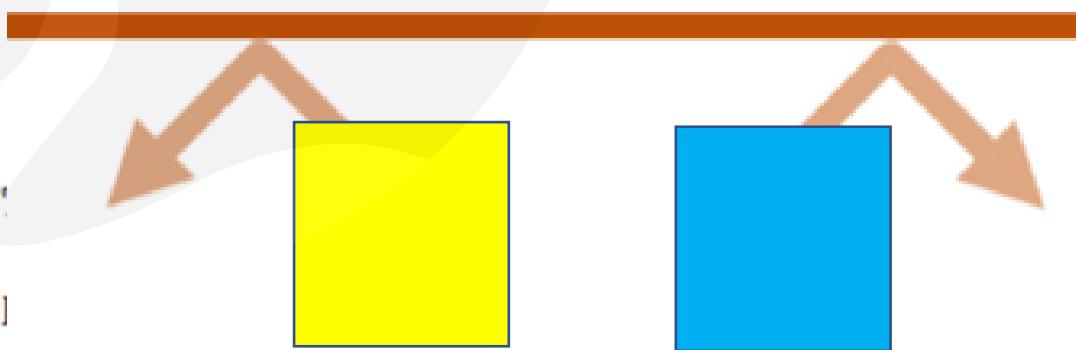


برخورد جعبه

اکنون باید بررسی کنیم که آیا جعبه از لبه پنجره گذشته یا نه. اگر گذشته، باید جعبه را تغییر جهت بدھیم تا در تکرار بعدی حلقه بازی جعبه در جهت جدید حرکت کند. این باعث می‌شود به نظر برسد که جعبه به لبه پنجره خورده و برگشته است.

مثلاً اگر جعبه به لبه بالایی برخورد کند، بسته به جهتی که جعبه هنگام برخورد بالبه دارد، به سمت پایین و چپ یا پایین و راست تغییر جهت می‌دهد.

```
if b['rect'].top < 0:  
    if b['dir'] == UPLEFT:  
        b['dir'] = DOWNLEFT  
    if b['dir'] == UPRIGHT:  
        b['dir'] = DOWNRIGHT
```



برخورد جعبه

```
if b['rect'].bottom > WINDOWHEIGHT:  
    if b['dir'] == DOWNLEFT:  
        b['dir'] = UPLEFT  
    if b['dir'] == DOWNRIGHT:  
        b['dir'] = UPRIGHT
```

به همین ترتیب این شرط برای همه جهت‌ها
تعریف می‌شود.

```
if b['rect'].left < 0:  
    if b['dir'] == DOWNLEFT:  
        b['dir'] = DOWNRIGHT  
    if b['dir'] == UPLEFT:  
        b['dir'] = UPRIGHT
```

```
if b['rect'].right > WINDOWWIDTH:  
    if b['dir'] == DOWNRIGHT:  
        b['dir'] = DOWNLEFT  
    if b['dir'] == UPRIGHT:  
        b['dir'] = UPLEFT
```

ترسیم جعبه ها روی پنجره در موقعیت های جدیدشان



هر بار که جعبه ها حرکت می کنند، باید با فراخوانی تابع آنها را در موقعیت های جدید خود در پنجره

قرار دهیم:

```
pygame.draw.rect(windowSurface, b['color'], b['rect'])
```

همانطور که می دانیم `b['rect']` و `b['color']` موقعیت و رنگ مستطیل است.

کد بالا آخرین خط حلقه `for` است.

کشیدن پنجره روی صفحه

پس از حلقه for، باید برای ترسیم پنجره روی صفحه، pygame.display.update() را فراخوانی کنیم:

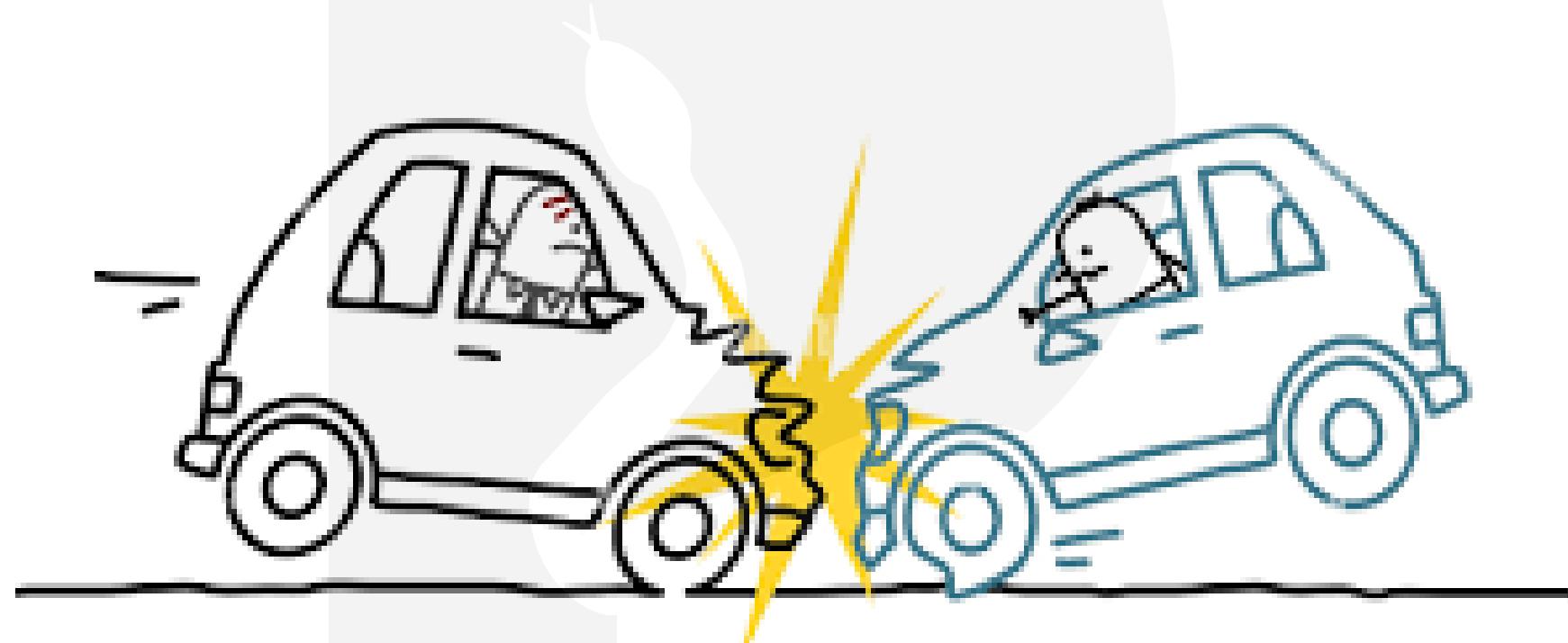
```
pygame.display.update()
```

```
time.sleep(0.02)
```



کامپیووتر می تواند آنقدر سریع حرکت کند، برخورد کند و جعبه ها را بکشد که اگر برنامه با سرعت کامل اجرا شود، تمام جعبه ها تار به نظر می رسند. به منظور اینکه برنامه را به آرامی اجرا کنیم تا بتوانیم جعبه ها را ببینیم time.sleep() را به برنامه می افزاییم. با فراخوانی time.sleep(0.02) برنامه به مدت ۰.۰۲ ثانیه یا ۲۰ میلی ثانیه بین هر کدام از حرکات جعبه ها مکث می کند. پس از این خط، اجرا به شروع حلقه بازی باز می گردد و روند را دوباره از نو آغاز می کند. به این ترتیب جعبه ها مدام در حال حرکت هستند، به دیوارها برخورد می کنند، و در موقعیت جدیدشان روی صفحه کشیده می شوند.

pygame ،، بـخورـ



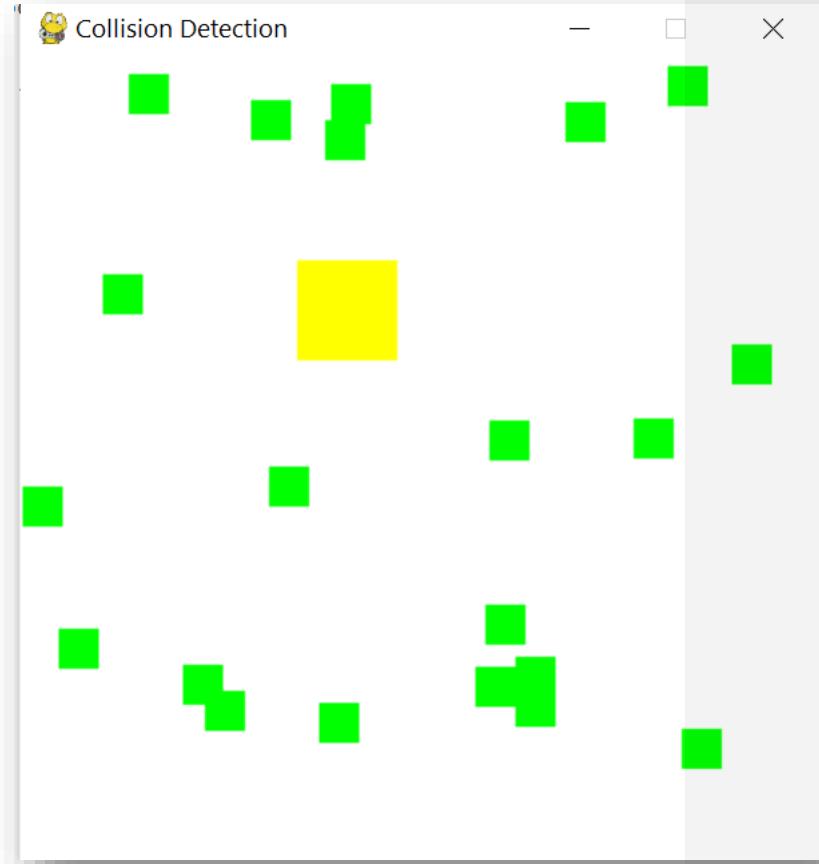
تشخیص برخورد



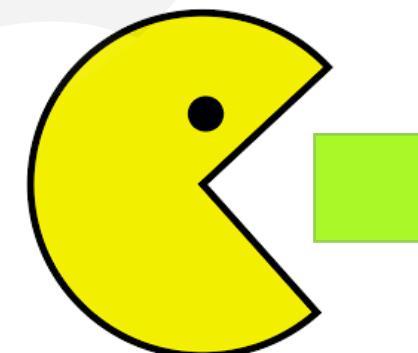
تشخیص برخورد شامل دریافتن این است که دو جسم روی صفحه یکدیگر را لمس می کنند. در بازی ما، تشخیص برخورد تعیین می کند که آیا دو مستطیل روی هم همپوشانی دارند یا خیر.

همچنین بررسی خواهیم کرد که چگونه برنامه های pygame می توانند از بازیکن از طریق صفحه کلید و ماوس ورودی دریافت کنند. این دو مفهوم بازی های شما را هیجان انگیز تر می کند!

برنامه تشخیص برخورد



در این برنامه، بازیکن از کلیدهای جهت دار صفحه کلید برای جابجایی جعبه زرد رنگ در صفحه نمایش استفاده می کند. مربع های سبز رنگ کوچکتری که نشان دهنده غذا هستند بر روی صفحه نمایش ظاهر می شوند و جعبه با برخورد با مربع ها، آنها را "می خورد".



کد تشخیص برخورد

```
import pygame, sys, random
from pygame.locals import *

# Set up pygame.
pygame.init()
mainClock = pygame.time.Clock()

# Set up the window.
WINDOWWIDTH = 400
WINDOWHEIGHT = 400
windowSurface = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT), 0, 32)
pygame.display.set_caption('Collision Detection')

# Set up the colors.
YELLOW = (255, 255, 0)
GREEN = (0, 255, 0)
WHITE = (255, 255, 255)

# Set up the player and food data structures.
foodCounter = 0
NEWFOOD = 40
FOODSIZE = 20
player = pygame.Rect(300, 100, 50, 50)
foods = []
for i in range(20):
    foods.append(pygame.Rect(random.randint(0, WINDOWWIDTH - FOODSIZE), ran
```

کد تشخیص برخورد

```
# Set up movement variables.  
moveLeft = False  
moveRight = False  
moveUp = False  
moveDown = False  
  
MOVESPEED = 6  
  
  
# Run the game loop.  
while True:  
    # Check for events.  
    for event in pygame.event.get():  
        if event.type == QUIT:  
            pygame.quit()  
            sys.exit()  
        if event.type == KEYDOWN:  
            # Change the keyboard variables.  
            if event.key == K_LEFT or event.key == K_a:  
                moveRight = False  
                moveLeft = True  
            if event.key == K_RIGHT or event.key == K_d:  
                moveLeft = False  
                moveRight = True
```

کد تشخیص برخورد

```
if event.key == K_UP or event.key == K_w:  
    moveDown = False  
    moveUp = True  
if event.key == K_DOWN or event.key == K_s:  
    moveUp = False  
    moveDown = True  
if event.type == KEYUP:  
    if event.key == K_ESCAPE:  
        pygame.quit()  
        sys.exit()  
    if event.key == K_LEFT or event.key == K_a:  
        moveLeft = False  
    if event.key == K_RIGHT or event.key == K_d:  
        moveRight = False  
    if event.key == K_UP or event.key == K_w:  
        moveUp = False  
    if event.key == K_DOWN or event.key == K_s:  
        moveDown = False  
  
foodCounter += 1  
if foodCounter >= NEWFOOD:  
    # Add new food.  
    foodCounter = 0  
    foods.append(pygame.Rect(random.randint(0, WINDOWWIDTH - FOODSIZE), random.randint(0,  
        WINDOWHEIGHT - FOODSIZE), FOODSIZE, FOODSIZE))
```

کد تشخیص برخورد

```
# Draw the white background onto the surface.  
windowSurface.fill(WHITE)  
  
# Move the player.  
if moveDown and player.bottom < WINDOWHEIGHT:  
    player.top += MOVESPEED  
if moveUp and player.top > 0:  
    player.top -= MOVESPEED  
if moveLeft and player.left > 0:  
    player.left -= MOVESPEED  
if moveRight and player.right < WINDOWWIDTH:  
    player.right += MOVESPEED  
  
# Draw the player onto the surface.  
pygame.draw.rect(windowSurface, YELLOW, player)  
  
# Check whether the player has intersected with any food squares.  
for food in foods[:]:  
    if player.colliderect(food):  
        foods.remove(food)  
  
# Draw the food.  
for i in range(len(foods)):  
    pygame.draw.rect(windowSurface, GREEN, foods[i])  
  
# Draw the window onto the screen.  
pygame.display.update()  
mainClock.tick(40)
```

شروع نوشتن کد

برنامه تشخیص برخورد pygame از همان مازولهایی که در برنامه اینیمیشن استفاده شد، به علاوه مازول رندوم استفاده می کند:

```
import pygame, sys, random  
from pygame.locals import *
```



پس از آن مقداردهی اولیه pygame را انجام میدهیم:

```
pygame.init()
```

استفاده از ساعت برای سرعت برنامه

در برنامه انیمیشن فراخوانی `time.sleep(0.02)` سرعت برنامه را کاهش می دهد تا برنامه خیلی سریع اجرا نشود. از آنجایی که این تابع برای همه کامپیوترها یک زمان را در نظر می گیرد، سرعت بقیه برنامه بستگی به سرعت کامپیوتر دارد. اگر بخواهیم برنامه ما در همه کامپیوترها یک سرعت را داشته باشد باید تابعی داشته باشیم که در کامپیوترهای سریعتر کندتر و در کامپیوترهای کند تر با سرعت بیشتر عمل کند. تابع زیر این کار را برای ما انجام میدهد:

```
mainClock = pygame.time.Clock()
```

متدهای `mainClock.tick()` که در حلقه وارد می شود سرعت اجرای بازی را کنترل می کند.



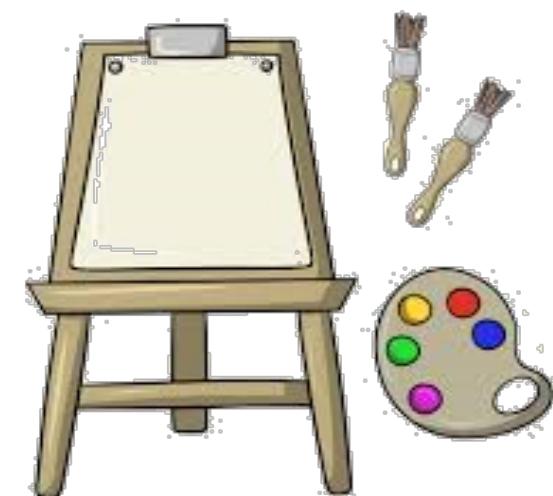
تنظیمات اولیه

پس از آن مشابه برنامه انیمیشن، تنظیم عرض و ارتفاع پنجره، و تعریف متغیرهای رنگ و مسیر را انجام میدهیم:

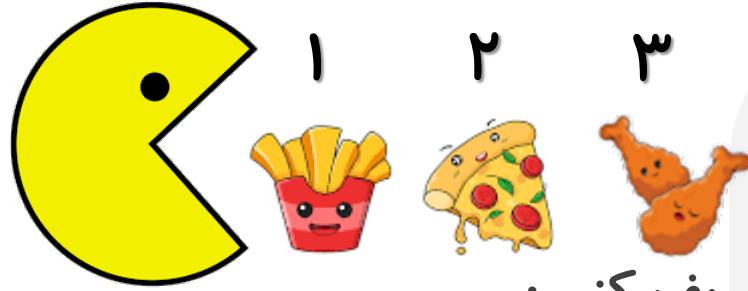
```
WINDOWWIDTH = 400  
WINDOWHEIGHT = 400
```

```
windowSurface = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT), 0, 32)  
pygame.display.set_caption('Collision Detection')
```

```
YELLOW = (255, 255, 0)  
GREEN = (0, 255, 0)  
WHITE = (255, 255, 255)
```



تنظیمات اولیه



حالا برای مربع های غدایی که در صفحه ظاهر می شوند تعدادی متغیر تعریف کنیم:

```
foodCounter = 0
```

```
NEWFOOD = 40
```

```
FOODSIZE = 20
```

متغیر player جعبه بازیکن است که موقعیت و اندازه آن به این صورت تعریف می شود:

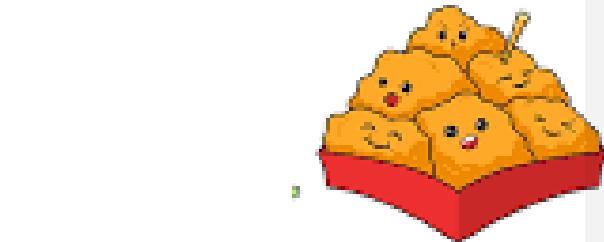
```
player = pygame.Rect(300, 100, 50, 50)
```

جعبه بازیکن مشابه جعبه های برنامه اینیمیشن حرکت می کند اما در این برنامه بازیکن می تواند حرکت جعبه کنترل کند.

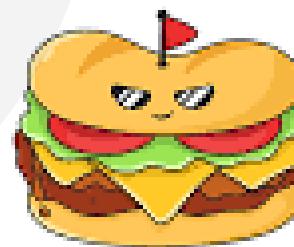
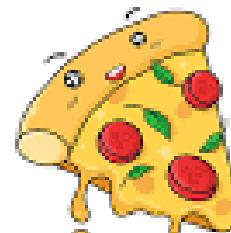
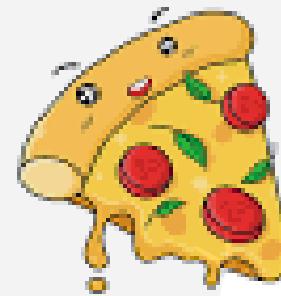
مربع های غذا

سپس مربع های غذا را ایجاد می کنیم:

```
foods = []
for i in range(20):
    foods.append(pygame.Rect(random.randint(0, WINDOWWIDTH - FOODSIZE), random.randint(0,
    WINDOWHEIGHT - FOODSIZE), FOODSIZE, FOODSIZE))
```



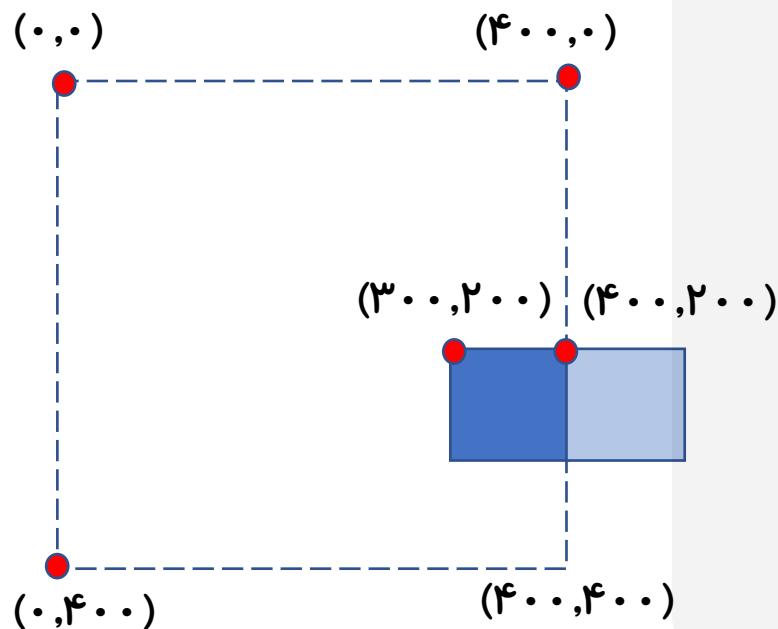
این کد هر مربع غذا را در لیست foods ذخیره می کند. حلقه for، 20 مربع غذا ایجاد کرده و بصورت تصادفی در صفحه قرار می دهد.



مربع های غذا

جديد نشان دهنده موقعیت و اندازه هر مربع غذای جدید است:

```
pygame.Rect(random.randint(0, WINDOWWIDTH - FOODSIZE))
```



دو پارامتر اول pygame.Rect() مختصات x و y گوش سمت چپ بالا هستند. مختصات تصادفی بین ۰ و اندازه پنجره منهای اندازه مربع غذا است. اگر مختصات تصادفی را بین ۰ و اندازه پنجره تنظیم کنید، مربع غذا ممکن است به طور کامل بیرون از پنجره قرار بگیرد، مانند شکل: سومین و چهارمین پارامتر pygame.Rect() عرض و ارتفاع مربع غذا هستند. هم عرض و هم ارتفاع مقادیر موجود در ثابت FOODSIZE هستند.

تنظیم متغیرهای حرکت

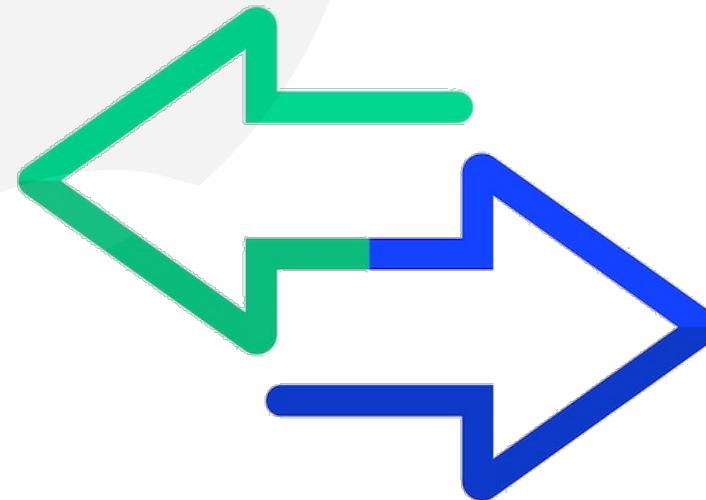
متغیرهای زیر برای هر جهتی که جعبه می تواند حرکت کند، مقدار اولیه مشخص می کنند. همهی این متغیرها دارای مقادیر Boolean هستند و همه در ابتدا روی False تنظیم می شوند. مثلاً وقتی بازیکن برای جابجایی جعبه، کلید جهت‌نمای چپ روی صفحه کلید خود را فشار می دهد، moveLeft روی True تنظیم شده است. وقتی کلید رها شود، moveLeft به حالت اولیه False باز می گردد.

```
moveLeft = False
```

```
moveRight = False
```

```
moveUp = False
```

```
moveDown = False
```



تنظیم متغیرهای حرکت

در ادامه حلقه‌ی بازی را می‌نویسیم و مشخص می‌کنیم که وقتی بازیکن از برنامه خارج شد چه باید بکنیم. از آنجا که کدهای مربوط به این قسمت مشابه فصل قبل است از توضیح دوباره آن خودداری می‌کنیم.

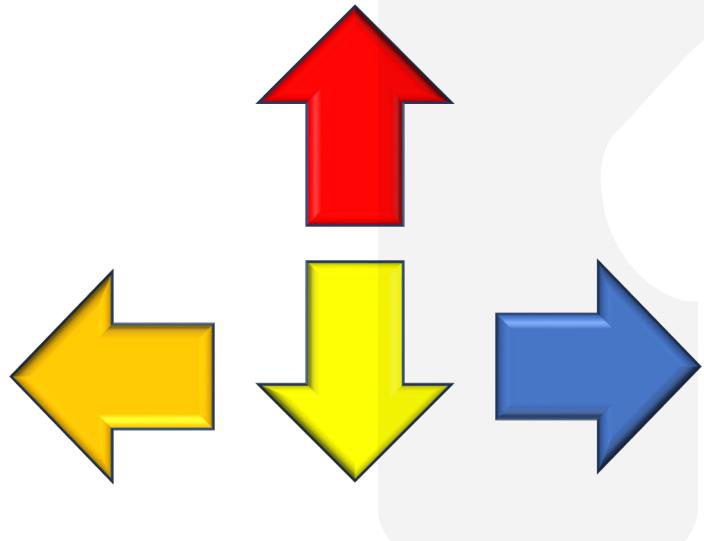
```
MOVESPEED = 6
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
```



کنترل رویداد KEYDOWN

ماژول pygame می‌تواند در پاسخ به ورودی کاربر از نوع ماوس یا صفحه کلید رویدادهایی را تولید کند. KEYDOWN رویدادی است که توسط pygame.event.get() بازگردانده شوند.

اگر نوع رویداد KEYDOWN باشد و بازیکن یک کلید جهت نما یا یک کلید WASD را فشار دهد، انتظار داریم جعبه حرکت کند. از دستور if برای بررسی کلید فشار داده شده استفاده می‌کنیم تا بفهمیم که جعبه باید در کدام جهت حرکت کند.

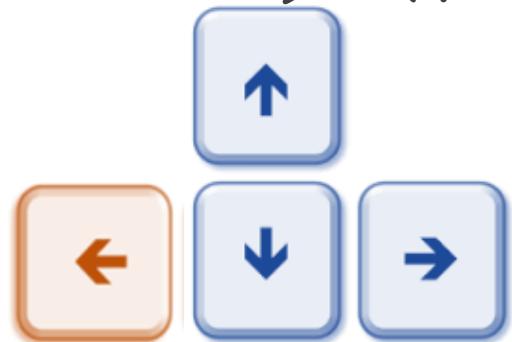


کنترل رویداد KEYDOWN

ماژول pygame می‌تواند در پاسخ به ورودی کاربر از نوع ماوس یا صفحه کلید رویدادهایی را تولید کند. رویدادی است که توسط pygame.event.get() بازگردانده شوند. اگر نوع رویداد KEYDOWN باشد یعنی بازیکن یک کلید جهت نما یا یک کلید WASD را فشار داده و انتظار داریم جعبه حرکت کند. از دستور if برای بررسی کلید فشار داده شده استفاده می‌کنیم تا بفهمیم که جعبه باید در کدام جهت حرکت کند.

```
if event.type == KEYDOWN:  
    if event.key == K_LEFT or event.key == K_a:  
        moveRight = False  
        moveLeft = True
```

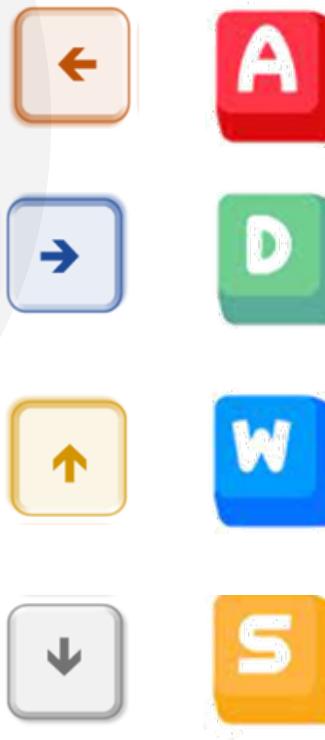
WASD K_LEFT و K_a به ترتیب نشان‌دهنده کلید جهت‌نمای چپ صفحه کلید و A در هستند.



کنترل رویداد KEYDOWN

به همین ترتیب این شرط برای همه جهت‌ها تعریف می‌شود.

```
if event.key == K_LEFT or event.key == K_a:  
    moveRight = False  
    moveLeft = True  
if event.key == K_RIGHT or event.key == K_d:  
    moveLeft = False  
    moveRight = True  
if event.key == K_UP or event.key == K_w:  
    moveDown = False  
    moveUp = True  
if event.key == K_DOWN or event.key == K_s:  
    moveUp = False  
    moveDown = True
```



کنترل رویداد KEYUP

وقتی بازیکن کلیدی را که فشار داده رها می‌کند، یک رویداد KEYUP تولید می‌شود:

```
if event.type == KEYUP:
```

اگر کلیدی که بازیکن رها کرد esc بود، پایتون باید برنامه را خاتمه دهد. به یاد داشته باشید، در pygame باید تابع sys.exit() را قبل از فراخوانی تابع pygame.quit() فراخوانی کنید:

```
if event.key == K_ESCAPE:
```

```
    pygame.quit()
```

```
    sys.exit()
```



کنترل رویداد KEYUP

کدهای زیر متغیر حرکت را در صورتی که کلید یک جهت رها شده باشد، روی False تنظیم می کند:

```
if event.key == K_LEFT or event.key == K_a:  
    moveLeft = False
```



```
if event.key == K_RIGHT or event.key == K_d:  
    moveRight = False
```



```
if event.key == K_UP or event.key == K_w:  
    moveUp = False
```



```
if event.key == K_DOWN or event.key == K_s:  
    moveDown = False
```

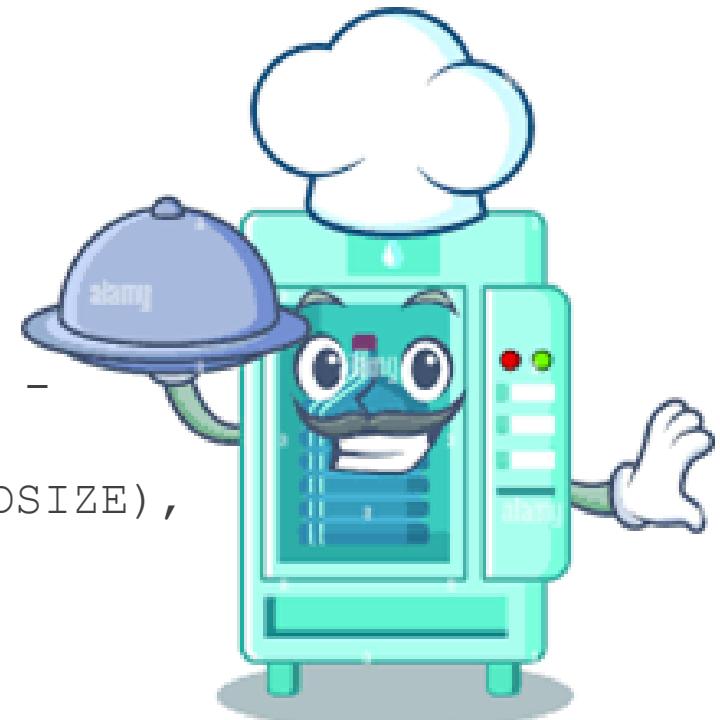


تنظیم متغیر حرکت به False از طریق رویداد KEYUP باعث می شود حرکت جعبه متوقف شود.

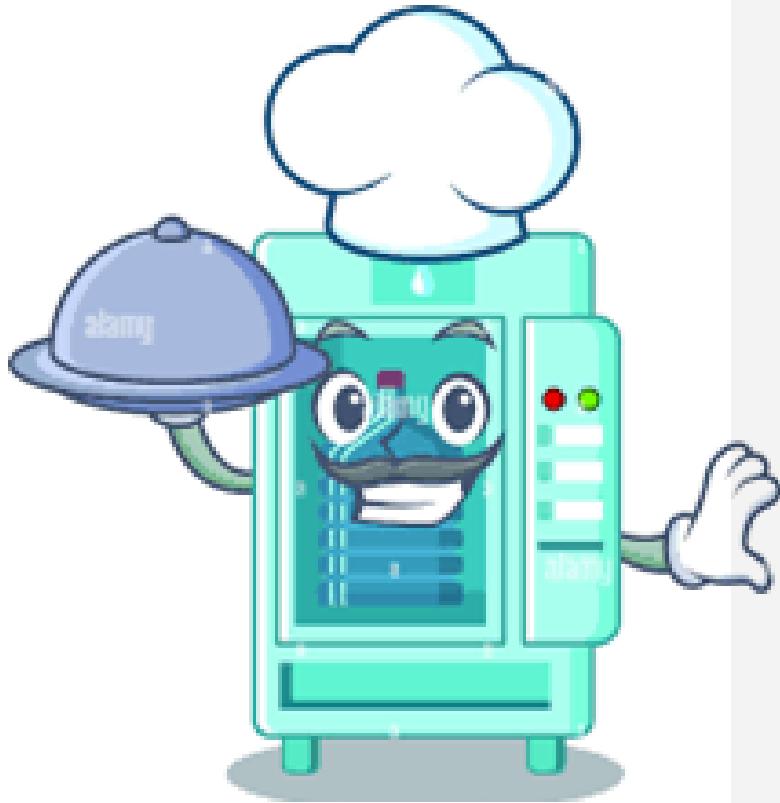
افزودن مربع های غذایی جدید

مربع های غذا به طور خودکار از طریق کد زیر تولید می شوند:

```
foodCounter += 1  
  
if foodCounter >= NEWFOOD:  
  
    foodCounter = 0  
  
    foods.append(pygame.Rect(random.randint(0, WINDOWWIDTH -  
        FOODSIZE), random.randint(0, WINDOWHEIGHT - FOODSIZE),  
        FOODSIZE, FOODSIZE))
```



افزودن مربع های غذایی جدید



متغیر foodCounter تعداد دفعات افزودن غذا را پیگیری می کند.
هر بار که حلقه بازی تکرار می شود، foodCounter یک عدد
افزایش می یابد.

هنگامی که foodCounter بزرگتر یا مساوی با ثابت
foodCounter شد، تنظیم مجدد می شود و یک
مربع غذای جدید توسط خط بعدی ایجاد می شود.

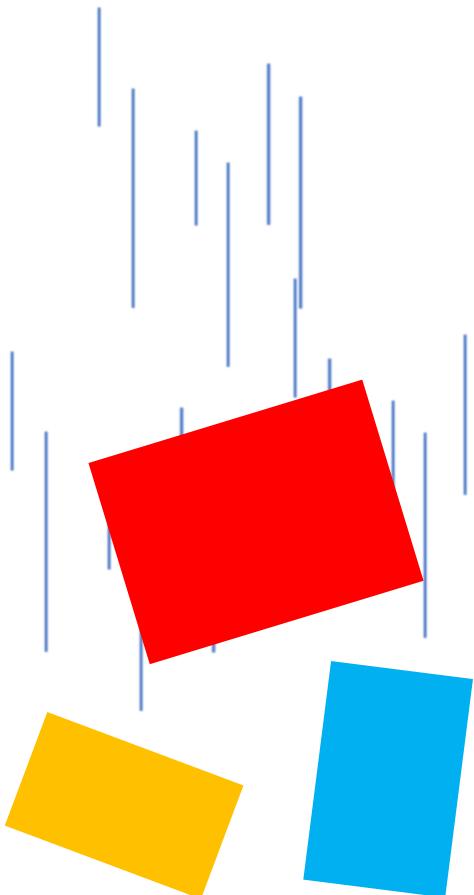
شما می توانید با تنظیم NEWFOOD، سرعت اضافه شدن مربع
های غذای جدید را تغییر دهید.

حرکت بازیکن درون پنجره

ما متغیرهای حرکت moveRight، moveLeft، moveUp، moveDown را بسته به اینکه بازیکن چه کلیدهایی را فشار داده باشد روی True یا False تنظیم کرده ایم. حالا ما باید جعبه بازیکن را که با pygame.Rect نشان داده شده و در ذخیره شده است، حرکت دهیم. این کار را با تنظیم مختصات x و y برای player انجام خواهیم داد:

```
if moveDown and player.bottom < WINDOWHEIGHT:  
    player.top += MOVESPEED
```

اگر moveDown روی True تنظیم شده باشد (و پایین جعبه بازیکن زیر لبه پایین پنجره نباشد)، خط player.top += MOVESPEED با افزودن top فعالی بازیکن، جعبه بازیکن را به پایین حرکت می دهد.



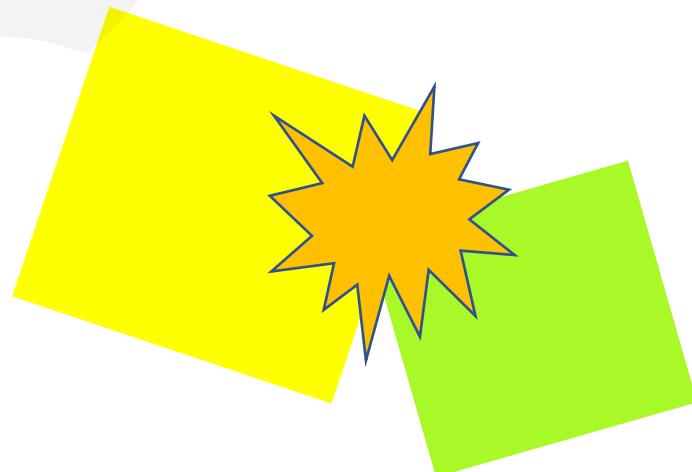
ترسیم بازیکن روی پنجره و بررسی برخورد

خط زیر جعبه بازیکن را روی پنجره ترسیم می کند:

```
pygame.draw.rect(windowSurface, YELLOW, player)
```

قبل از ترسیم مربع های غذا روی صفحه، باید ببینیم که جعبه بازیکن با هر یک از مربع ها برخورد می کند یا نه. در صورت برخورد، آن مربع باید از لیست غذاها حذف شود. از متده تشخصیص برخورد collidrect() برای این کار استفاده می کنیم:

```
for food in foods[:]:  
    if player.colliderect(food):  
        foods.remove(food)
```



ترسیم مربع های غذا روی پنجره

کدهای زیر مشابه کدی است که برای رسم جعبه بازیکن استفاده کردیم:

```
for i in range(len(foods)):  
    pygame.draw.rect(windowSurface, GREEN, foods[i])
```

پایان برنامه

حالا که مربع های بازیکن و غذا روی صفحه هستند، پنجره آماده به روزرسانی است، بنابراین متدهای update() و tick() را در خط بعد فراخوانی کرده و برنامه با فراخوانی متدهای Clock.tick() و Clock.update() را اجرا کنید.

ایجاد کردیم پایان می دهیم:

```
pygame.display.update()  
  
mainClock.tick(40)
```

این برنامه از طریق حلقه بازی ادامه می یابد و به روز رسانی ادامه می یابد تا زمانی که بازیکن خارج شود.



صبا و تحميله ,, pygame



استفاده از صدای و تصاویر

در فصل‌های قبل یاد گرفتید که چگونه برای ساختن برنامه‌های رابط کاربری گرافیکی که دارای گرافیک هستند ورودی را از طریق صفحه کلید یا ماوس بگیرید. همچنین یاد گرفتید که چگونه اشکال مختلف را بکشید. در این فصل نحوه اضافه کردن صدا، موسیقی و تصویر را به بازی یاد خواهید گرفت.

همه فایلهای مربوط به این قسمت در سایت بارگذاری شده است:

<http://pythonteeek.com/files/pygame>



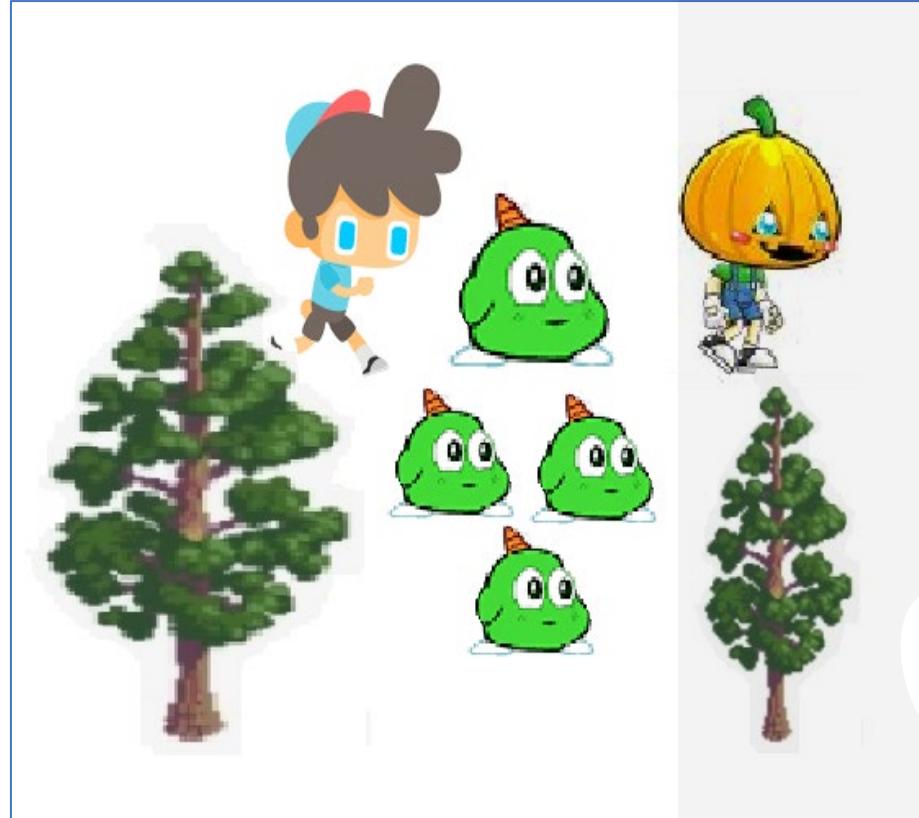
اسپرایت (sprite)

اسپرایت یک تصویر دو بعدی است که به عنوان بخشی از گرافیک روی یک صفحه نمایش استفاده می شود.

شکل زیر چند نمونه اسپرایت را نشان می دهد.



اسپرایت (sprite)



تصاویر اسپرایت در روی پس زمینه کشیده می شوند. می توانید تصویر اسپرایت را به صورت افقی در بیاورید. همچنین می توانید یک تصویر اسپرایت را چندین بار روی یک پنجره بکشید و همچنین اندازه اسپرایت ها بزرگتر یا کوچکتر از تصویر اسپرایت اصلی کنید. تصویر پس زمینه را نیز می توان یک اسپرایت بزرگ در نظر گرفت. شکل زیر نشان می دهد که چگونه اسپرایت ها با هم استفاده می شوند.

فایل های صدا و تصویر

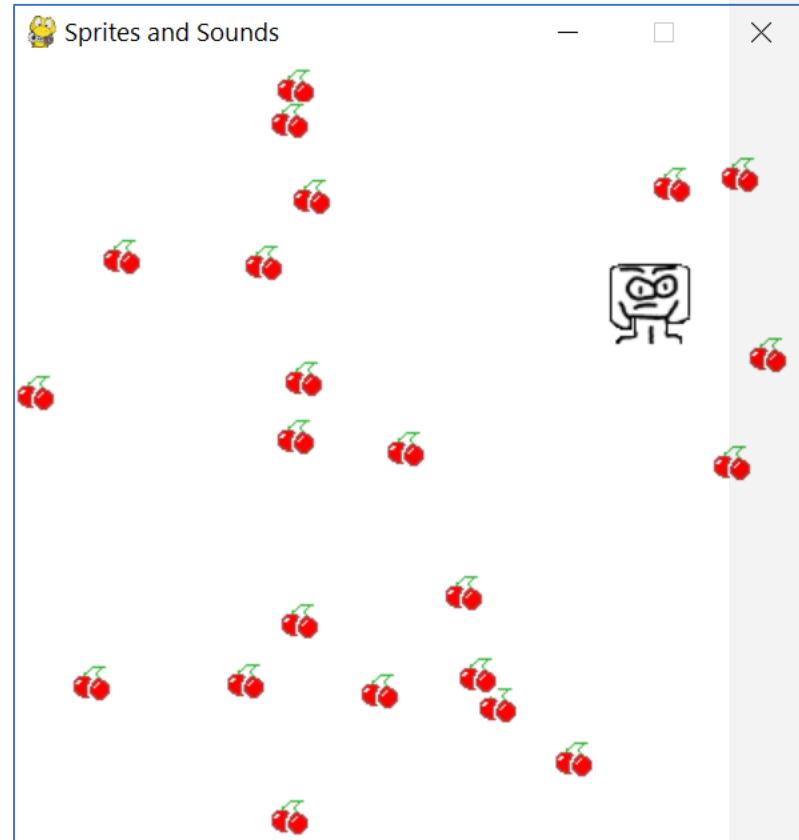
اسپرایت ها در فایل های تصویری در رایانه شما ذخیره می شوند. چندین فرمت تصویر وجود دارد که می تواند استفاده کند. . فرمت تصویر در pygame شامل BMP، PNG، JPG و GIF است.

شما باید فایل تصویر را در همان پوشه فایل py که برنامه پایتون شما در آن است قرار دهید.

فرمت های فایل صوتی که pygame پشتیبانی می کند MIDI، WAV و MP3 هستند.



برنامه اسپرایت و صدا



برنامه این فصل همان برنامه تشخیص برخورد از فصل قبل است. با این حال، در این برنامه به جای استفاده از مربع هایی با ظاهر ساده از اسپرایت ها استفاده می کنیم. همچنین به جای جعبه از یک شخص برای نشان دادن بازیکن و یک گیلاس به جای مربع غذای سبز استفاده خواهیم کرد. همچنین موسیقی پس زمینه پخش خواهیم کرد و وقتی اسپرایت بازیکن یکی از گیلاس ها را می خورد یک جلوه صوتی اضافه خواهیم کرد.

در این بازی، اسپرایت بازیکن، اسپرایت های گیلاس را می خورد و همانطور که گیلاس می خورد، رشد خواهد کرد.

کد اسپرایت و صدا

```
import pygame, sys, time, random
from pygame.locals import *

# Set up pygame.
pygame.init()
mainClock = pygame.time.Clock()

# Set up the window.
WINDOWWIDTH = 400
WINDOWHEIGHT = 400
windowSurface = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT), 0, 32)
pygame.display.set_caption('Sprites and Sounds')

# Set up the colors.
WHITE = (255, 255, 255)

# Set up the block data structure.
player = pygame.Rect(300, 100, 40, 40)
playerImage = pygame.image.load('player.png')
playerStretchedImage = pygame.transform.scale(playerImage, (40, 40))
foodImage = pygame.image.load('cherry.png')
foods = []
for i in range(20):
    foods.append(pygame.Rect(random.randint(0, WINDOWWIDTH - 20), random.randint(0, WINDOWHEIGHT - 20), 20, 20))

foodCounter = 0
NEWFOOD = 40
```

کد اسپرایت و صدا

```
# Set up keyboard variables.  
moveLeft = False  
moveRight = False  
moveUp = False  
moveDown = False  
  
MOVESPEED = 6  
  
# Set up the music.  
pickUpSound = pygame.mixer.Sound('pickup.wav')  
pygame.mixer.music.load('background.mid')  
pygame.mixer.music.play(-1, 0.0)  
musicPlaying = True  
  
# Run the game loop.  
while True:  
    # Check for the QUIT event.  
    for event in pygame.event.get():  
        if event.type == QUIT:  
            pygame.quit()  
            sys.exit()  
        if event.type == KEYDOWN:  
            # Change the keyboard variables.  
            if event.key == K_LEFT or event.key == K_a:  
                moveRight = False  
                moveLeft = True  
            if event.key == K_RIGHT or event.key == K_d:  
                moveLeft = False  
                moveRight = True
```

کد اسپرایت و صدا

```
if event.key == K_UP or event.key == K_w:
    moveDown = False
    moveUp = True
if event.key == K_DOWN or event.key == K_s:
    moveUp = False
    moveDown = True
if event.type == KEYUP:
    if event.key == K_ESCAPE:
        pygame.quit()
        sys.exit()
    if event.key == K_LEFT or event.key == K_a:
        moveLeft = False
    if event.key == K_RIGHT or event.key == K_d:
        moveRight = False
    if event.key == K_UP or event.key == K_w:
        moveUp = False
    if event.key == K_DOWN or event.key == K_s:
        moveDown = False
    if event.key == K_x:
        player.top = random.randint(0, WINDOWHEIGHT - player.height)
        player.left = random.randint(0, WINDOWWIDTH - player.width)
if event.key == K_m:
    if musicPlaying:
        pygame.mixer.music.stop()
    else:
        pygame.mixer.music.play(-1, 0.0)
musicPlaying = not musicPlaying
```

کد اسپرایت و صدا

```
if event.type == MOUSEBUTTONUP:  
    foods.append(pygame.Rect(event.pos[0] - 10, event.pos[1] - 10, 20, 20))  
foodCounter += 1  
if foodCounter >= NEWFOOD:  
    # Add new food.  
    foodCounter = 0  
    foods.append(pygame.Rect(random.randint(0, WINDOWWIDTH - 20), random.randint(0, WINDOWHEIGHT - 20), 20, 20))  
  
# Draw the white background onto the surface.  
windowSurface.fill(WHITE)  
  
# Move the player.  
if moveDown and player.bottom < WINDOWHEIGHT:  
    player.top += MOVESPEED  
if moveUp and player.top > 0:  
    player.top -= MOVESPEED  
if moveLeft and player.left > 0:  
    player.left -= MOVESPEED  
if moveRight and player.right < WINDOWWIDTH:  
    player.right += MOVESPEED  
  
# Draw the block onto the surface.  
windowSurface.blit(playerStretchedImage, player)
```

کد اسپرایت و صدا

```
# Check whether the block has intersected with any food squares.  
for food in foods[:]:  
    if player.colliderect(food):  
        foods.remove(food)  
        player = pygame.Rect(player.left, player.top, player.width + 2, player.height + 2)  
        playerStretchedImage = pygame.transform.scale(playerImage, (player.width, player.height))  
        if musicPlaying:  
            pickUpSound.play()  
  
    # Draw the food.  
    for food in foods:  
        windowSurface.blit(foodImage, food)  
  
    # Draw the window onto the screen.  
    pygame.display.update()  
    mainClock.tick(40)
```

راه اندازی برنامه

بیشتر کدهای این برنامه همان برنامه Collision Detection فصل قبل است. ما فقط بر روی قسمت هایی تمرکز خواهیم کرد که اسپرایت ها و صدا اضافه می شوند.

```
import pygame, sys, time, random
from pygame.locals import *
pygame.init()
mainClock = pygame.time.Clock()
WINDOWWIDTH = 400
WINDOWHEIGHT = 400
windowSurface = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT), 0, 32)
pygame.display.set_caption('Sprites and Sounds')
```

افزودن اسپرایت

بر خلاف برنامه های قبلی که فقط یک متغیر بازیکن را نشان می داد اینجا از سه متغیر استفاده خواهیم کرد:

متغیر اول بازیکن که یک Rect را ذخیره می کند که مکان و اندازه بازیکن را مشخص می کند. این متغیر شامل تصویر بازیکن نیست. بازیکن در ابتدای برنامه، گوشه بالا و سمت چپ در (300,100) واقع شده است، و دارای ارتفاع و عرض اولیه 40 پیکسل است:

```
player = pygame.Rect(300, 100, 40, 40)
```

دومین متغیری که بازیکن را نشان می دهد playerImage نام فایل تصویر pygame.image.load() است. تابع Surface است که دارای تصویری است که روی سطح آن کشیده شده است. ما این شیء Surface را در داخل playerImage ذخیره می کنیم.

```
playerImage = pygame.image.load('player.png')
```



افزودن اسپرایت

متغیر سوم یک تابع جدید در ماژول pygame.transform است. تابع `pygame.transform.scale()` می‌تواند یک اسپرایت را کوچک یا بزرگ کند.

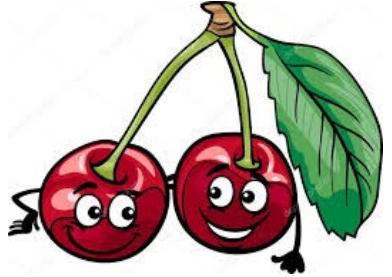
```
playerStretchedImage = pygame.transform.scale(playerImage, (40, 40))
```

اولین آرگومان یک شی Surface است که تصویر روی آن کشیده شده است. آرگومان دوم یک تاپل برای عرض و ارتفاع جدید تصویر است. تابع `pygame.transform.scale()` یک شی Surface را به همراه تصویر با اندازه جدید برمی‌گرداند.

در برنامه این فصل، اسپرایت بازیکن را بزرگتر می‌کنیم تا گیلاس بیشتری بخورد. ما تصویر اصلی را در متغیر `playerImage` ذخیره می‌کنیم و تصویر بزرگ شده را در متغیر `playerStretchedImage` ذخیره می‌کنیم.



افزودن اسپرایت



در خط بعد، مجدداً `load()` Surface را فراخوانی می کنیم تا یک شی `cherry` ایجاد کنیم. مطمئن شوید که فایل های `cherry.png` و `player.png` را در `pygame` همان پوشه فایل `spritesAndSounds.py` دارید. در غیر این صورت، نمی تواند آنها را پیدا کند و خطا می دهد.

```
foodImage = pygame.image.load('cherry.png')
```

تنظیم موسیقی و صدا



در مرحله بعد باید فایل های صوتی را بارگذاری کنیم. دو مژول در `pygame` برای صدا وجود دارد. مژول `pygame.mixer` که می تواند افکت های صوتی کوتاهی را در طول مدت بازی پخش کند و مژول `pygame.mixer.music` که می تواند موسیقی پس زمینه را پخش کند.

افزودن فایل های صوتی

برای ایجاد شیء Sound تابع pygame.mixer.Sound () را فراخوانی می کنیم.

```
pickUpSound = pygame.mixer.Sound('pickup.wav')
```

```
pygame.mixer.music.load('background.mid')
```

```
pygame.mixer.music.play(-1, 0.0)
```

کد () pygame.mixer.music.load() برای بارگیری موسیقی پس زمینه فراخوانی می شود.

با کد () pygame.mixer.music.play() افکت صوتی شروع به پخش می کند. اولین پارامتر به pygame می گوید که چند بار بعد از اولین باری که موسیقی پس زمینه را پخش می کنیم آن را تکرار کند. بنابراین عدد ۵ باعث می شود pygame شش بار موسیقی پس زمینه را پخش کند. در اینجا پارامتر ۱ - یک حالت خاص است که باعث می شود موسیقی پس زمینه برای همیشه تکرار شود.

افزودن فایل های صوتی

```
pygame.mixer.music.play(-1, 0.0)
```

```
musicPlaying = True
```

دومین پارامتر `play()` نقطه شروع اجرای فایل صوتی است. مقدار `0.0` برحسب ثانیه است و نشان می

دهد که موسیقی پس زمینه از ابتدا پخش می شود.

در نهایت، متغیر `musicPlaying` دارای یک مقدار `Boolean` است که به برنامه می گوید که آیا باید موسیقی پس زمینه و افکت های صوتی را پخش کند یا خیر. این خوب است که به بازیکن این امکان را بدهید که برنامه را بدون پخش صدا اجرا کند.

روشن و خاموش کردن صدا



کلید M موسیقی پس زمینه را روشن یا خاموش می کند. اگر MusicPlaying روی True تنظیم شده باشد، موسیقی پس زمینه در حال پخش است و ما باید آن را با فراخوانی pygame.mixer.music.stop() متوقف کنیم. اگر MusicPlaying روی False تنظیم شده باشد، موسیقی پس زمینه در حال حاضر پخش نمی شود، و باید آن را با فراخوانی play() پخش کنیم. خطوط زیر از دستور if برای این کار

```
if event.key == K_m:  
    if musicPlaying:  
        pygame.mixer.music.stop()  
  
    else:  
        pygame.mixer.music.play(-1, 0.0)  
  
    musicPlaying = not musicPlaying
```

استفاده می کنند:

روشن و خاموش کردن صدا



چه موسیقی در حال پخش باشد یا نه، ما می خواهیم مقدار musicPlaying را تغییر دهیم. تغییر دادن یک مقدار بولین به معنای تنظیم یک مقدار خلاف مقدار فعلی آن است. خط musicPlaying = not musicPlaying متغیر را به این صورت تغییر می دهد که اگر در حال حاضر True است به False و اگر در حال حاضر False است آن را روی True تنظیم می کند.

ترسیم بازیکن روی پنجره

خط زیر اسپرایت بازیکن را با استفاده از () blit() روی شی Surface پنجره می کشد

```
windowSurface.blit(playerStretchedImage, player)
```

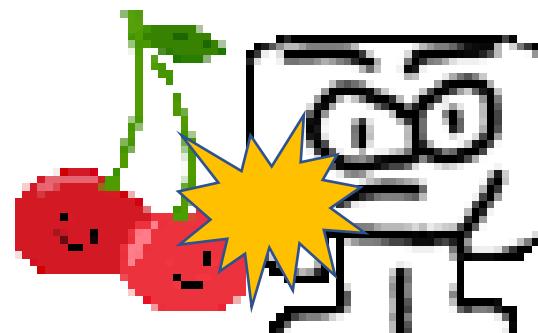


اولین پارامتر تابع، اسپرایت بازیکن و دومی موقعیت بازیکن است.

بررسی برخورد

این کد مشابه کد برنامه های قبلی است، اما چند خط جدید در آن وجود دارد:

```
if player.colliderect(food):
    foods.remove(food)
    player = pygame.Rect(player.left, player.top, player.width + 2,
player.height + 2)
    playerStretchedImage = pygame.transform.scale(playerImage,
(player.width, player.height))
    if musicPlaying:
        pickUpSound.play()
```

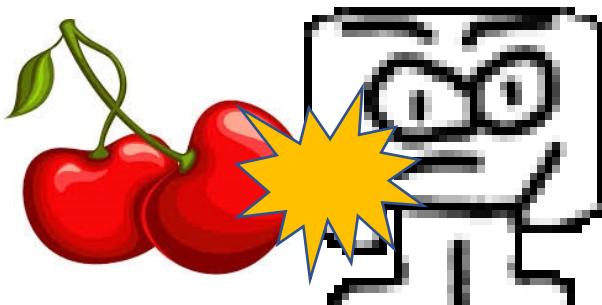


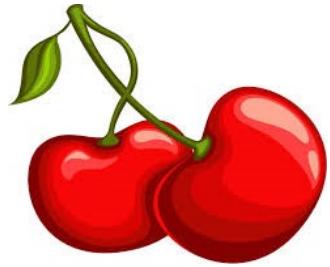
بررسی برخورد

وقتی اسپرایت بازیکن یکی از گیلاس ها را می خورد، اندازه آن دو پیکسل در ارتفاع و عرض افزایش می یابد. در این حالت یک Rect جدید که دو پیکسل بزرگتر از Rect قدیمی است جایگزین می شود.

Rect نشان دهنده موقعیت و اندازه بازیکن است و از سوی دیگر تصویر بازیکن در یک playerStretchedImage یک تصویر کشیده شده جدید ایجاد می شود.

در خط آخر در صورتی که موزیک روی MusicPlaying تنظیم شده باشد (که به این معنی است که صدای روشن است)، متدهنگ برخورد را پخش می کند.





کشیدن گیلاس روی پنجره

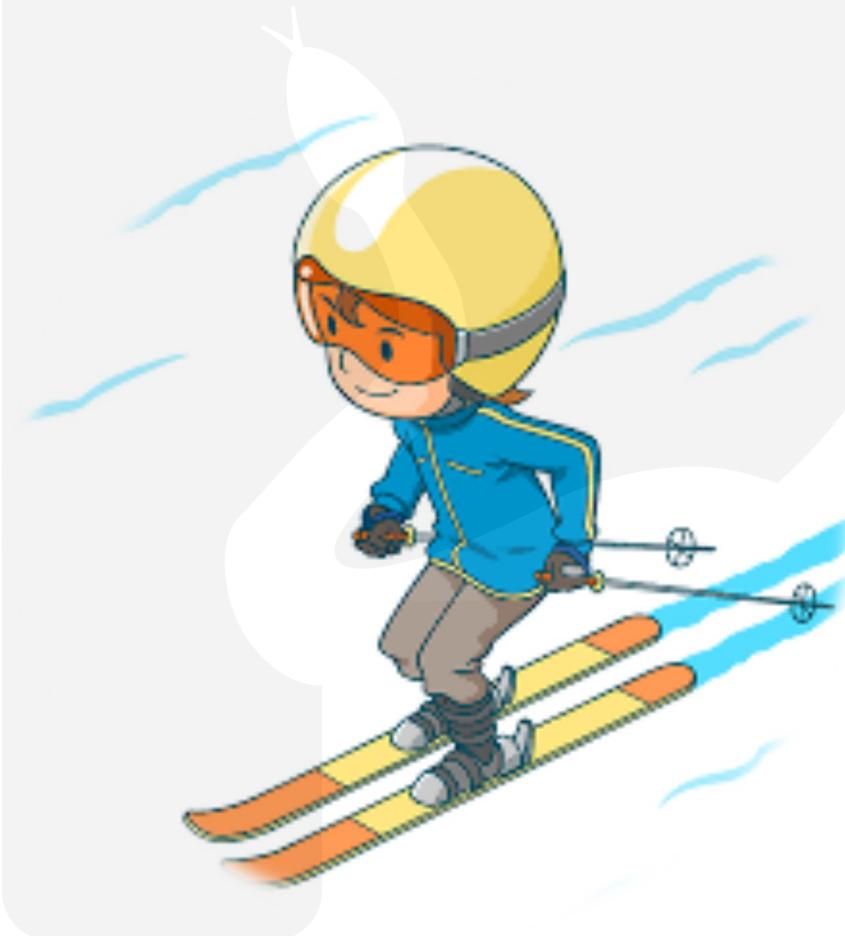
در برنامه های قبلی تابع `pygame.draw.rect()` را برای ترسیم فراخوانی کردید تا یک مربع سبز برای هر `Rect` ذخیره شده در لیست غذاها بکشد. در این برنامه، شما می خواهید به جای آن، اسپرایت های گیلاس را بکشید. متدهای `blit()` برای تصویر گیلاس روی آن کشیده شده است را فراخوانی کنید:

```
for food in foods:  
    windowSurface.blit(foodImage, food)
```

متغیر `food` که شامل هر یک از اشیاء `Rect` در `foods` موجود در هر تکرار از حلقه `for` است، به متدهای `blit()` می گوید که عکس غذا را کجا رسم کند.

اکنون می دانیم چگونه یک پنجره بسازیم، اسپرایت ها را نمایش دهیم، ورودی های صفحه کلید و ماوس را جمع آوری کنیم، صداها را پخش کنیم و تشخیص برخورد را پیاده سازی کنیم. حالا آماده ساخت یک بازی گرافیکی در `pygame` هستیم.

Ski Tree Game



Ski Tree



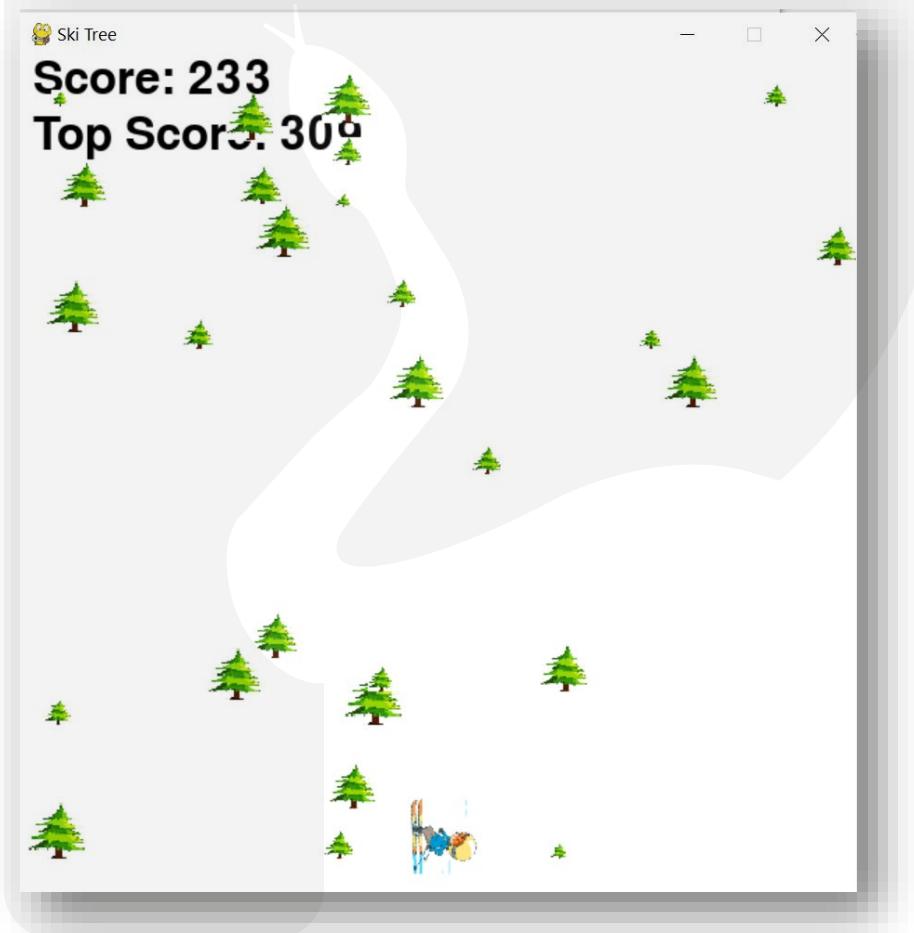
در فصلهای قبل به مژول pygame این موضوع پرداختیم و نشان دادیم که چگونه می توان از امکانات فراوان آن استفاده کرد. در این فصل، ما از این آموخته ها برای ایجاد یک بازی گرافیکی به نام Ski Tree استفاده می کنیم. در بازی Ski Tree، بازیکن یک اسپرایت (شخصیت بازیکن) را کنترل می کند که باید از بین درختان که از بالای صفحه نمایش سقوط می کند عبور کند. بازیکن هر چقدر مدت زمان بیشتری بتواند به رد شدن از درختان ادامه دهد، امتیاز بالاتری خواهد گرفت.

همه فایلهای مربوط به این قسمت در سایت بارگذاری شده است:

<http://pythontreek.com/files/pygame>

Ski Tree

هنگامی که این برنامه را اجرا می کنید، بازی مانند شکل زیر خواهد بود.



Ski Tree ↵

```
import pygame, random, sys
from pygame.locals import *

WINDOWWIDTH = 600
WINDOWHEIGHT = 600
TEXTCOLOR = (0, 0, 0)
BACKGROUNDCOLOR = (255, 255, 255)
FPS = 60
TREEMINSIZE = 10
TREEMAXSIZE = 40
TREEMINSPEED = 1
TREEMAXSPEED = 8
ADDNEWTREERATE = 6
PLAYERMOVERATE = 5

def terminate():
    pygame.quit()
    sys.exit()

def waitForPlayerToPressKey():
    while True:
        for event in pygame.event.get():
            if event.type == QUIT:
                terminate()
            if event.type == KEYDOWN:
                if event.key == K_ESCAPE: # Pressing ESC quits.
                    terminate()
            return
```



Ski Tree ↴

```
def playerHasHitTree(playerRect, trees):
    for t in trees:
        if playerRect.colliderect(t['rect']):
            return True
    return False

def drawText(text, font, surface, x, y):
    textobj = font.render(text, 1, TEXTCOLOR)
    textrect = textobj.get_rect()
    textrect.topleft = (x, y)
    surface.blit(textobj, textrect)

# Set up pygame, the window, and the mouse cursor.
pygame.init()
mainClock = pygame.time.Clock()
windowSurface = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
pygame.display.set_caption('Ski Tree')
pygame.mouse.set_visible(False)

# Set up the fonts.
font = pygame.font.SysFont(None, 48)

# Set up sounds.
gameOverSound = pygame.mixer.Sound('gameover.wav')
pygame.mixer.music.load('background.mid')
```

Ski Tree

```
# Set up images.
playerImage = pygame.image.load('skier.png')
playerRect = playerImage.get_rect()
treeImage = pygame.image.load('tree.png')

# Show the "Start" screen.
windowSurface.fill(BACKGROUNDCOLOR)
drawText('Ski Tree', font, windowSurface, (WINDOWWIDTH / 3), (WINDOWHEIGHT / 3))
drawText('Press any key to start.', font, windowSurface, (WINDOWWIDTH / 3) - 30, (WINDOWHEIGHT / 3) + 50)
pygame.display.update()
waitForPlayerToPressKey()

topScore = 0
while True:
    # Set up the start of the game.
    trees = []
    score = 0
    playerRect.topleft = (WINDOWWIDTH / 2, WINDOWHEIGHT - 50)
    moveLeft = moveRight = moveUp = moveDown = False
    reverseCheat = slowCheat = False
    treeAddCounter = 0
    pygame.mixer.music.play(-1, 0.0)
```

Ski Tree ک

```
while True: # The game loop runs while the game part is playing.  
    score += 1 # Increase score.  
  
    for event in pygame.event.get():  
        if event.type == QUIT:  
            terminate()  
        if event.type == KEYDOWN:  
            if event.key == K_LEFT or event.key == K_a:  
                moveRight = False  
                moveLeft = True  
            if event.key == K_RIGHT or event.key == K_d:  
                moveLeft = False  
                moveRight = True  
            if event.key == K_UP or event.key == K_w:  
                moveDown = False  
                moveUp = True  
            if event.key == K_DOWN or event.key == K_s:  
                moveUp = False  
                moveDown = True
```

Ski Tree ↵

```
if event.type == KEYUP:
    if event.key == K_ESCAPE:
        terminate()
    if event.key == K_LEFT or event.key == K_a:
        moveLeft = False
    if event.key == K_RIGHT or event.key == K_d:
        moveRight = False
    if event.key == K_UP or event.key == K_w:
        moveUp = False
    if event.key == K_DOWN or event.key == K_s:
        moveDown = False

if event.type == MOUSEMOTION:
    # If the mouse moves, move the player where to the cursor.
    playerRect.centerx = event.pos[0]
    playerRect.centery = event.pos[1]
# Add new trees at the top of the screen, if needed.

treeAddCounter += 1
if treeAddCounter == ADDNEWTREERATE:
    treeAddCounter = 0
    treeSize = random.randint(TREEMINSIZE, TREEMAXSIZE)
    newTree = {'rect': pygame.Rect(random.randint(0, WINDOWWIDTH - treeSize), 0 - treeSize, treeSize, treeSize),
               'speed': random.randint(TREEMINSPEED, TREEMAXSPEED),
               'surface':pygame.transform.scale(treeImage, (treeSize, treeSize)),
              }
    trees.append(newTree)
```

Ski Tree ↵

```
# Move the player around.  
if moveLeft and playerRect.left > 0:  
    playerRect.move_ip(-1 * PLAYEROVERATE, 0)  
if moveRight and playerRect.right < WINDOWWIDTH:  
    playerRect.move_ip(PLAYEROVERATE, 0)  
if moveUp and playerRect.top > 0:  
    playerRect.move_ip(0, -1 * PLAYEROVERATE)  
if moveDown and playerRect.bottom < WINDOWHEIGHT:  
    playerRect.move_ip(0, PLAYEROVERATE)  
  
# Move the trees down.  
for t in trees:  
    t['rect'].move_ip(0, t['speed'])  
  
# Delete trees that have fallen past the bottom.  
for t in trees[:]:  
    if t['rect'].top > WINDOWHEIGHT:  
        trees.remove(t)  
  
# Draw the game world on the window.  
windowSurface.fill(BACKGROUNDCOLOR)  
  
# Draw the score and top score.  
drawText('Score: %s' % (score), font, windowSurface, 10, 0)  
drawText('Top Score: %s' % (topScore), font, windowSurface, 10, 40)
```

Ski Tree ↵

```
# Draw the player's rectangle.  
windowSurface.blit(playerImage, playerRect)  
  
# Draw each tree.  
for t in trees:  
    windowSurface.blit(t['surface'], t['rect'])  
  
pygame.display.update()  
  
# Check if any of the trees have hit the player.  
if playerHasHitTree(playerRect, trees):  
    if score > topScore:  
        topScore = score # set new top score  
    break  
  
mainClock.tick(FPS)  
  
# Stop the game and show the "Game Over" screen.  
pygame.mixer.music.stop()  
gameOverSound.play()  
  
drawText('GAME OVER', font, windowSurface, (WINDOWWIDTH / 3), (WINDOWHEIGHT / 3))  
drawText('Press a key to play again.', font, windowSurface, (WINDOWWIDTH / 3) - 80, (WINDOWHEIGHT / 3) + 50)  
pygame.display.update()  
waitForPlayerToPressKey()  
  
gameOverSound.stop()
```

وارد کردن ماثول ها

بازی Ski Tree همان ماثولهای قبلی pygame را وارد می‌کند:

```
import pygame, sys, time, random  
from pygame.locals import *
```



تنظیم متغیرهای ثابت

خطوط زیر ثابت هایی را برای ابعاد پنجره، رنگ متن و رنگ پس زمینه تعریف می کند:

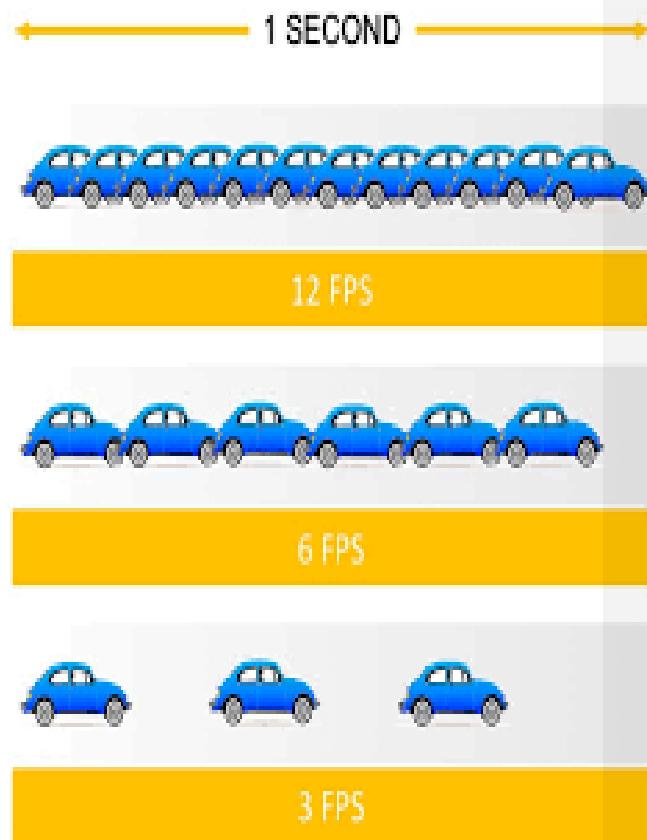
```
WINDOWWIDTH = 600
```

```
WINDOWHEIGHT = 600
```

```
TEXTCOLOR = (0, 0, 0)
```

```
BACKGROUNDCOLOR = (255, 255, 255)
```

تنظیم متغیرهای ثابت



در خط زیر، شما ثابت FPS را برای تعداد فریم در هر ثانیه بازی تنظیم می کنید:

$$\text{FPS} = 60$$

فریم صفحه‌ای است که در یک بار تکرار در حلقه بازی ترسیم می‌شود.

شما FPS را در متدهای mainClock.tick()، در خطوط پایانی کد، استفاده می‌کنید تا تابع بداند چه مدت باید برنامه را متوقف کند. در اینجا FPS روی ۶۰ تنظیم شده است، اما می‌توانید FPS را به مقدار بالاتریا کمتر تغییر دهید تا بازی سریعتر یا کندتر اجرا شود.

تنظیم متغیرهای ثابت

خطوط زیر متغیرهای ثابت بیشتری را برای درختان در حال پایین آمدن در صفحه تنظیم می کند:

TREEMINSIZE = 10

TREEMAXSIZE = 40

TREEMINSPEED = 1

TREEMAXSPEED = 8

عرض و ارتفاع درختان ها بین TREEMINSIZE و TREEMAXSIZE نمایش بین TREEMAXSPEED و TREEMINSPEED پیکسل در هر تکرار از حلقه بازی خواهد بود.



تنظیم متغیرهای ثابت

ADDNEWTREERATE = 6

PLAYERTOVERATE = 5



در هر ADDNEWTREERATE مرتبه تکرار در طی حلقه بازی یک درخت جدید به بالای پنجره اضافه می شود.

تعداد پیکسل هایی است که PLAYERTOVERATE کاراکتر بازیکن در هر تکرار از حلقه بازی حرکت می کند.

تعریف توابع

چندین تابع وجود دارد که شما برای این بازی ایجاد خواهید کرد. تابع terminate() و () wateForPlayerToPressKey به ترتیب، بازی را به پایان می رساند و متوقف می کند، تابع drawText() برخورد بازیکن را با درختان بررسی می کند، و تابع playerHasHitTree() امتیاز و سایر متن ها را در صفحه نمایش می دهد.

def function_name():

پایان بازی:

در مازول pygame باید هم sys.exit() را فراخوانی کنیم و هم pygame.quit() را در تابع terminate() قرار می دهیم:

```
def terminate():
    pygame.quit()
    sys.exit()
```

اکنون فقط باید به جای pygame.quit() تابع sys.exit() را فراخوانی کنید.



توقف بازی:

گاهی اوقات می خواهید برنامه را تا زمانی که بازیکن یک کلید را فشار دهد، متوقف کنید، مانند همان ابتدای بازی که متن عنوان بازی ظاهر می شود یا در پایان زمانی که Game Over را نشان می دهد:

```
def waitForPlayerToPressKey():
    while True:
        for event in pygame.event.get():
            if event.type == QUIT:
                terminate()
            if event.type == KEYDOWN:
                if event.key == K_ESCAPE:
                    terminate()
    return
```



توقف بازی:

اگر بازیکن پنجره را بسته باشد در حالی که برنامه منتظر است که بازیکن یک کلید را فشار دهد، pygame یک رویداد QUIT ایجاد می کند. اگر بازیکن از بازی خارج شده باشد، پایتون terminate() را فراخوانی می کند.

اگر بازی یک رویداد KEYDOWN دریافت کرد، ابتدا باید بررسی کند که esc فشرده شده یا نه.

اگر بازیکن esc را فشار داد، برنامه باید خاتمه یابد.

در غیر این صورت، اجرا از این if می گذرد و برنامه از تابع waitForPlayerToPressKey() خارج می شود. اگر رویداد KEYDOWN یا QUIT ایجاد نشود، کد به چرخش ادامه می دهد. در این حالت تا زمانی که بازیکن کلیدی را فشار دهد، حلقه هیچ کاری انجام نمی دهد.



بررسی برخورد

در صورتی که کاراکتر بازیکن با یکی از درختان برخورد کند، تابع `playerHasHitTree()` مقدار `True` را

برمی‌گرداند:

```
def playerHasHitTree(playerRect, trees):  
    for t in trees:  
        if playerRect.colliderect(t['rect']):  
            return True  
  
    return False
```



کشیدن متن در پنجره

رسم متن روی پنجره شامل چند مرحله است که ما آن را با `drawText()` انجام می دهیم. به این ترتیب، زمانی که می خواهیم امتیاز بازیکن یا متن Game Over را روی صفحه نمایش دهیم، تنها یک تابع برای فراخوانی وجود دارد.

```
def drawText(text, font, surface, x, y):  
    textobj = font.render(text, 1, TEXTCOLOR)  
    textrect = textobj.get_rect()  
    textrect.topleft = (x, y)  
    surface.blit(textobj, textrect)
```



کشیدن متن در پنجره

ابتدا، فراخوانی متد render() یک Surface ایجاد می کند که متن را با یک فونت خاص تبدیل به تصویر می کند.

در مرحله بعد، اندازه و مکان متن را با استفاده از get_rect() وارد می کنیم.

در نهایت، متن رender شده بر روی سطحی که در تابع drawText() وارد شده ترسیم می شود. نمایش متن در چند مرحله بیشتر از فراخوانی تابع print() دارد. ولی اگر این کد را در یک تابع به نام drawText() قرار دهید، برای نمایش متن روی صفحه فقط فقط باید این تابع را فراخوانی کنید.



راه اندازی pygame و راه اندازی پنجره

اکنون که متغیرها و توابع ثابت به پایان رسیده اند، شروع به فراخوانی توابع pygame می کنیم:

```
pygame.init()
```

مقداردهی اولیه:

```
mainClock = pygame.time.Clock()
```

جلوگیری از اجرای سریع برنامه:

```
windowSurface = pygame.display.set_mode( (WINDOWWIDTH, WINDOWHEIGHT) )
```

تنظیم ابعاد پنجره:

```
pygame.display.set_caption('Ski Tree')
```

عنوان بازی:

```
pygame.mouse.set_visible(False)
```

غیرقابل مشاهده کردن ماوس:

راه اندازی pygame و راه اندازی پنجره

اکنون که متغیرها و توابع ثابت به پایان رسیده اند، شروع به فراخوانی توابع pygame می کنیم:

```
font = pygame.font.SysFont(None, 48)
```

تنظیم فونت:

```
gameOverSound = pygame.mixer.Sound('gameover.wav')
```

تنظیم صدا:

```
pygame.mixer.music.load('background.mid')
```

تنظیم اسپرایت ها:

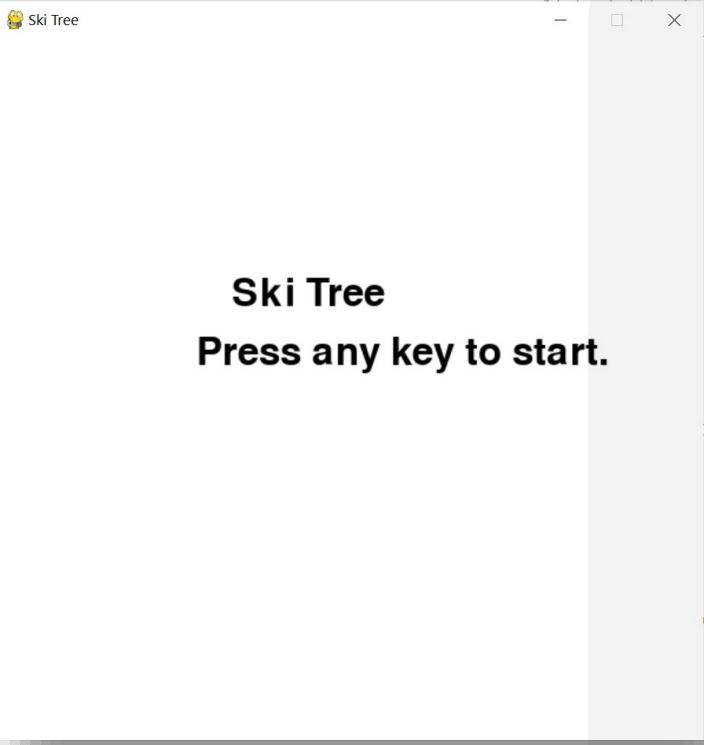
```
playerImage = pygame.image.load('skier.png')
```

```
playerRect = playerImage.get_rect()
```

```
treeImage = pygame.image.load('tree.png')
```



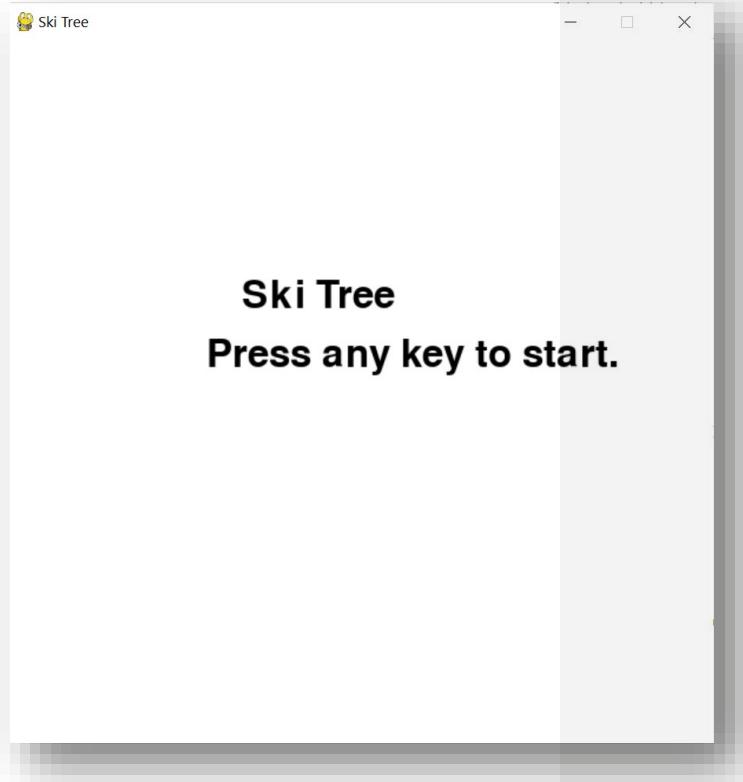
نمایش صفحه شروع



هنگامی که بازی برای اولین بار شروع می شود، پایتون باید عنوان Ski Tree را روی صفحه نمایش نشان دهد. همچنین باید به بازیکن بگویید که می تواند با فشار دادن هر کلیدی بازی را شروع کند. این صفحه بخاطر این ظاهر می شود که بازیکن پس از اجرای برنامه، زمانی برای آماده شدن داشته باشد. در ادامه، کدی را برای فراخوانی تابع drawText() می نویسیم:

```
windowSurface.fill(BACKGROUNDCOLOR)  
  
drawText('Ski Tree', font, windowSurface, (WINDOWWIDTH / 3), (WINDOWHEIGHT / 3))  
  
drawText('Press any key to start.', font, windowSurface, (WINDOWWIDTH / 3) - 30,  
(WINDOWHEIGHT / 3) + 50)
```

نمایش صفحه شروع



ما برای تابع `drawText` پنج آرگومان وارد می کنیم:

۱. رشته متنی که می خواهیم ظاهر شود
۲. فونتی که می خواهیم رشته با آن ظاهر شود
۳. سطحی که متن روی آن رندر می شود
۴. مختصات `X` بر روی سطحی که در آن متن رسم می شود
۵. مختصات `Y` روی سطحی که در آن متن رسم می شود

شروع بازی

اکنون با تمام توابع تعریف شده، می توانیم شروع به نوشتن کد اصلی بازی کنیم. در ادامه توابعی را که قبل از تعریف کردیم فراخوانی می کنیم.

مقدار متغیر topScore زمانی که برنامه برای اولین بار اجرا می شود از ۰ شروع می شود. هر زمان که بازیکن می بازد و امتیازی بزرگتر از امتیاز برتر فعلی دارد، امتیاز برتر با این امتیاز بزرگتر جایگزین می شود.

```
topScore = 0
```

```
while True:
```



حلقه بی نهایت بالا از نظر فنی حلقه اصلی بازی نیست. حلقه اصلی بازی رویدادها و ترسیم پنجره را در حین بازی مدیریت می کند. در عوض، این حلقه while هر بار که بازیکن یک بازی جدید را شروع می کند، تکرار می شود. زمانی که بازیکن می بازد و بازی ریست می شود، برنامه به خط بالا باز می گردد.

شروع بازی

لیست اولیه درختان:

امتیاز اولیه:

```
trees = []
```

```
score = 0
```

موقعیت اولیه بازیکن: playerRect.topleft = (WINDOWWIDTH / 2, WINDOWHEIGHT - 50)

متغیرهای ثابت برای حرکت: moveLeft = moveRight = moveUp = moveDown = False

```
treeAddCounter = 0
```

متغیر treeAddCounter شمارنده ای است که به برنامه می گوید چه زمانی باید یک درخت جدید در بالای صفحه اضافه کند. این مقدار در هر بار که حلقه بازی تکرار می شود، ۱ عدد افزایش می یابد. وقتی treeAddCounter صفر می شود و یک درخت جدید به بالای صفحه اضافه می شود.

```
pygame.mixer.music.play(-1, 0.0)
```

موسیقی پس زمینه:

حلقه بازی

کد حلقه بازی به طور مداوم وضعیت بازی را با تغییر موقعیت بازیکن و درخت ها، کنترل رویدادهای ایجاد شده توسط pygame و ترسیم بازی روی صفحه به روز می کند.

خط زیر شروع حلقه اصلی بازی است:

```
while True:
```

```
    score += 1
```

خط بالا امتیاز بازیکن را در هر تکرار از حلقه بازی افزایش می دهد. هر چه بازیکن بیشتر بتواند بدون باخت بازی کند، امتیاز او بالاتر خواهد بود. حلقه تنها زمانی خارج می شود که بازیکن بازی را ببازد یا از برنامه خارج شود.



کنترل رویدادهای صفحه کلید

چهار نوع رویداد وجود دارد که برنامه کنترل می کند: QUIT، KEYDOWN، KEYUP و .MOUSEMOTION

خط زیر شروع کد مدیریت رویداد است:

```
for event in pygame.event.get():
```

اگر type برابر با QUIT باشد، کاربر برنامه را بسته است. برنامه از بازی خارج می شود.

```
if event.type == QUIT:
```

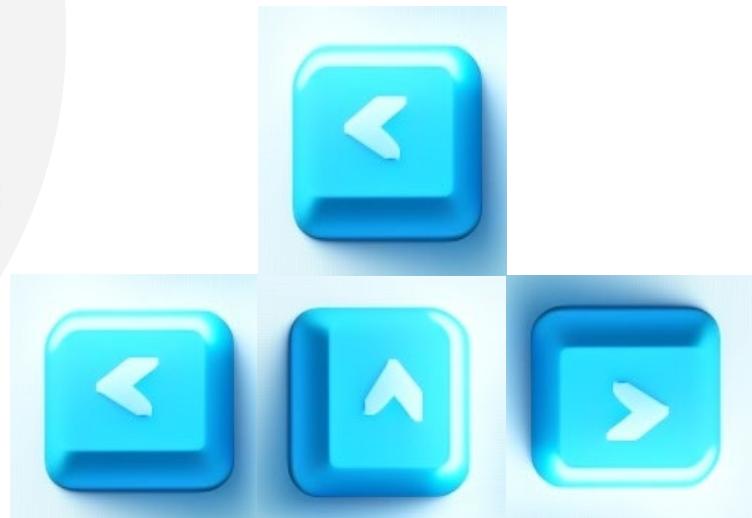
```
    terminate()
```



کنترل رویدادهای صفحه کلید

خطوط بعد بررسی می کنند که آیا رویداد با فشار دادن یکی از کلیدهای جهت دار یا WASD توسط بازیکن ایجاد شده است یا خیر. بسته به کلیدی که فشرده شده جهت حرکت مشخص می شود.

```
if event.type == KEYDOWN:  
    if event.key == K_LEFT or event.key == K_a:  
        moveRight = False  
        moveLeft = True  
    if event.key == K_RIGHT or event.key == K_d:  
        moveLeft = False  
        moveRight = True  
    if event.key == K_UP or event.key == K_w:  
        moveDown = False  
        moveUp = True  
    if event.key == K_DOWN or event.key == K_s:  
        moveUp = False  
        moveDown = True
```



کنترل رویدادهای صفحه کلید

در هر زمانی در طول بازی، بازیکن می تواند esc را برای خروج فشار دهد:

```
if event.key == K_ESCAPE:  
    terminate()
```

کد بالا تعیین می کند که آیا کلید رها شده esc بوده یا خیر. اگر چنین است، خط بعد تابع () را برای خروج از برنامه فراخوانی می کند.



کنترل رویدادهای صفحه کلید

خطوط بعد بررسی می کنند که آیا رویداد با رها کردن یکی از کلیدهای جهت دار یا WASD توسط بازیکن ایجاد شده است یا خیر. در آن صورت، کد متغیر حرکت مربوطه را روی False قرار می دهد.

```
if event.type == KEYUP:  
    if event.key == K_ESCAPE:  
        terminate()  
    if event.key == K_LEFT or event.key == K_a:  
        moveLeft = False  
    if event.key == K_RIGHT or event.key == K_d:  
        moveRight = False  
    if event.key == K_UP or event.key == K_w:  
        moveUp = False  
    if event.key == K_DOWN or event.key == K_s:  
        moveDown = False
```



کنترل حرکت ماوس

رویداد MOUSEMOTION با حرکت ماوس ایجاد می شود:

```
if event.type == MOUSEMOTION:  
    playerRect.centerx = event.pos[0]  
    playerRect.centery = event.pos[1]
```

ویژگی pos یک تاپل از مختصات x و y محل حرکت مکان نما ماوس در پنجره است.

اگر نوع رویداد MOUSEMOTION باشد، کاراکتر بازیکن به سمت موقعیت نشانگر ماوس حرکت می کند.

خطوط داخل شرط، مختصات مرکز x و y کاراکتر بازیکن را روی مختصات x و y نشانگر ماوس تنظیم می کنند.





اضافه کردن درختان جدید

در هر تکرار از حلقه بازی، کد متغیر treeAddCounter را یک واحد افزایش می دهد:

```
treeAddCounter += 1
```

وقتی treeAddCounter به مقدار ADDNEWTREERATE می رسد، زمان اضافه کردن یک درخت جدید به بالای صفحه است. اول، treeAddCounter را صفر می کنیم:

```
if treeAddCounter == ADDNEWTREERATE:  
    treeAddCounter = 0
```

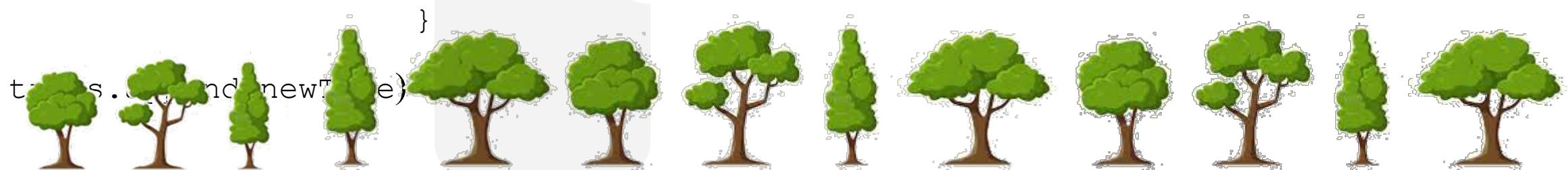
پس از اندازه درخت را بصورت پیکسل وارد می کنیم که یک عدد صحیح تصادفی بین TREEMINSIZE و TREEMAXSIZE خواهد بود.

```
treeSize = random.randint(TREEMINSIZE, TREEMAXSIZE)
```

اضافه کردن درختان جدید

ساختار داده برای درخت یک دیکشنری با کلیدهای "rect" ، "rect" ، "speed" و "surface" است. کلید "rect" مکان و اندازه درخت را مشخص می کند. کلید "speed" سرعت پایین آمدن درخت را مشخص می کند، و در نهایت کلید "surface" مکانی روی سطح که تصویر در آن قرار می گیرد را مشخص می کند.

```
newTree =  
    { 'rect': pygame.Rect(random.randint(0, WINDOWWIDTH - treeSize), 0 -  
        treeSize, treeSize, treeSize),  
        'speed': random.randint(TREEMINSPEED, TREEMAXSPEED),  
        'surface':pygame.transform.scale(treeImage, (treeSize, treeSize)),  
    }  
trees.append(newTree)
```



حرکت دادن کاراکتر بازیکن و درختان

چهار متغیر حرکت moveUp، moveDown، moveLeft و moveRight هنگامی که رویدادهای pygame KEYDOWN و KEYUP را تولید می کند، به ترتیب روی True و False تنظیم می شوند.

اگر بازیکن به سمت چپ حرکت کند و لبه چپ بازیکن بزرگتر از ۰ (که لبه سمت چپ پنجره است) باشد playerRect باید به سمت چپ حرکت کند:

```
if moveLeft and playerRect.left > 0:  
    playerRect.move_ip(-1 * PLAYEROVERATE, 0)
```

متده move_ip() مکان Rect را با تعدادی از پیکسلها جابجا می کند. اولین آرگومان برای move_ip() تعداد پیکسلها برای حرکت دادن Rect به راست است (برای انتقال آن به چپ، یک عدد صحیح منفی وارد می شود). آرگومان دوم تعداد پیکسلهایی است که Rect باید به پایین جابجا شود (برای بالا بردن آن، یک عدد صحیح منفی وارد می شود).

حرکت دادن کاراکتر بازیکن و درختان

خطوط بعدی همین کار را برای سه جهت دیگر انجام می دهند: راست، بالا و پایین:

```
if moveRight and playerRect.right < WINDOWWIDTH:  
    playerRect.move_ip(PLAYEROVERATE, 0)  
  
if moveUp and playerRect.top > 0:  
    playerRect.move_ip(0, -1 * PLAYEROVERATE)  
  
if moveDown and playerRect.bottom < WINDOWHEIGHT:  
    playerRect.move_ip(0, PLAYEROVERATE)
```

هر یک از سه دستور if در خطوط بالا بررسی می کند که متغیر حرکت روی True تنظیم شده و لبه Rect بازیکن داخل پنجره است. سپس move_ip() را برای جابجایی Rect فراخوانی می کند.

حرکت دادن کاراکتر بازیکن و درختان

اکنون کد روی هر tree در لیستی از trees برای پایین آوردن آنها حلقه می زند:

```
for t in trees:  
    t['rect'].move_ip(0, t['speed'])
```

هر درخت چند پیکسل به اندازه سرعت خود به پایین حرکت می کند.

حذف درختان

هر درخت که زیر لبه پایینی پنجره می رسد باید از لیست درختان حذف شود.

```
for t in trees[:]:  
    if t['rect'].top > WINDOWHEIGHT:  
        trees.remove(t)
```

به یاد داشته باشید که نباید آیتم های لیست را اضافه یا حذف کنید و در عین حال روی لیست حلقه تکرار داشته باشید. به

جای تکرار روی لیست درختان با حلقه for، از طریق یک کپی از لیست درختان([:] trees) تکرار کنید.

کشیدن پنجره

قبل از ترسیم هر چیز دیگری، خط زیر تمام صفحه را از هر چیزی که قبلاً روی آن کشیده شده بود پاک کردن می‌کند:

```
windowSurface.fill(BACKGROUNDCOLOR)
```

رسم امتیاز بازیکن

خطوط زیر متن را برای امتیاز فعلی و امتیاز برتر در گوش سمت چپ بالای پنجره نشان می‌دهد:

```
drawText('Score: %s' % (score), font, windowSurface, 10, 0)
```

```
drawText('Top Score: %s' % (topScore), font, windowSurface, 10, 40)
```

اعداد آخر مختصات x و y جایی است که متن باید قرار گیرد.





ترسیم کاراکتر بازیکن

متده `blit()` تصویر کاراکتر بازیکن را در `windowSurface` در مکان مشخص شده در `playerRect` ترسیم می کند:

```
windowSurface.blit(playerImage, playerRect)
```

ترسیم درخت

کد زیر در حلقه `for` هر درخت را در `windowSurface` ترسیم می کند:

```
for t in trees:
```

```
    windowSurface.blit(t['surface'], t['rect'])
```

اکنون که همه چیز به `windowSurface` کشیده شده است، باید صفحه را به روز کنیم تا بازیکن بتواند آنچه را که در آنجا وجود دارد ببیند:

```
pygame.display.update()
```

بررسی برخورد

خط زیر بررسی می کند که آیا بازیکن با درخت برخورد کرده است یا خیر. در صورتی که بازیکن با هر یک از درختان برخورد کرده باشد، این تابع True را برمی گرداند. در غیر این صورت، تابع False را برمی گرداند.

```
if playerHasHitTree(playerRect, trees):  
    if score > topScore:  
        topScore = score  
    break
```

اگر بازیکن به درخت برخورد کند و اگر امتیاز فعلی بالاتر از امتیاز برتر باشد، امتیاز برتر به روز می شود. این برنامه با خط break از حلقه بازی خارج می شود و به سمت پایان بازی می رود.



FPS

برای جلوگیری از اجرا شدن حلقه بازی با سرعت بالا (که برای بازیکن خیلی سریع است) تابع mainClock.tick() را فراخوانی می کنیم تا مکث بسیار کوتاهی در بازی ایجاد کند:

mainClock.tick(FPS)



این مکث به اندازه کافی طولانی خواهد بود تا اطمینان حاصل شود که حدود ۴۰ تکرار (مقدار ذخیره شده در متغیر FPS) در حلقه بازی در هر ثانیه اتفاق می افتد.

صفحه باخت



GAME OVER!

هنگامی که بازیکن بازنده می شود، بازی پخش موسیقی پس زمینه

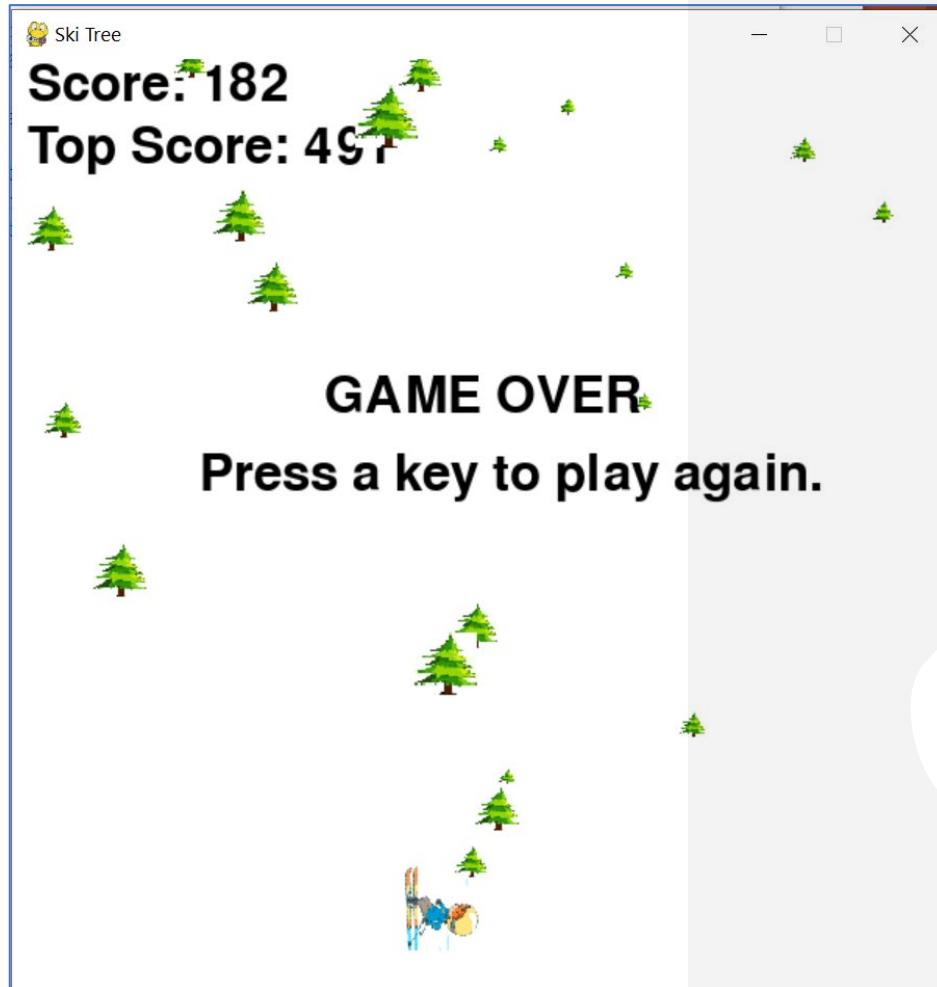
را متوقف می کند و جلوه صوتی "game over" را پخش می کند:

```
pygame.mixer.music.stop()  
gameOverSound.play()
```

سپس خطوط زیر تابع `drawText()` را برای رسم متن "game over" در `windowSurface` فراخوانی می کند:

```
drawText('GAME OVER', font, windowSurface, (WINDOWWIDTH / 3), (WINDOWHEIGHT / 3))  
  
drawText('Press a key to play again.', font, windowSurface, (WINDOWWIDTH / 3) - 80,  
(WINDOWHEIGHT / 3) + 50)
```

پایان بازی



```
pygame.display.update()
```

```
waitForPlayerToPressKey()
```

با نمایش متن، بازی به روز و سپس متوقف می شود تا زمانی که بازیکن با فراخوانی تابع `waitForPlayerToPressKey()`، کلیدی `game over` را فشار دهد. با فشار دادن یک کلید، افکت صوتی `game over` ممکن است همچنان وجود داشته باشد یا نباشد. برای توقف این جلوه صوتی قبل از شروع بازی جدید، کد زیر را می نویسیم:

```
gameOverSound.stop()
```

پایان بازی

```
pygame.display.update()  
  
waitForPlayerToPressKey()
```

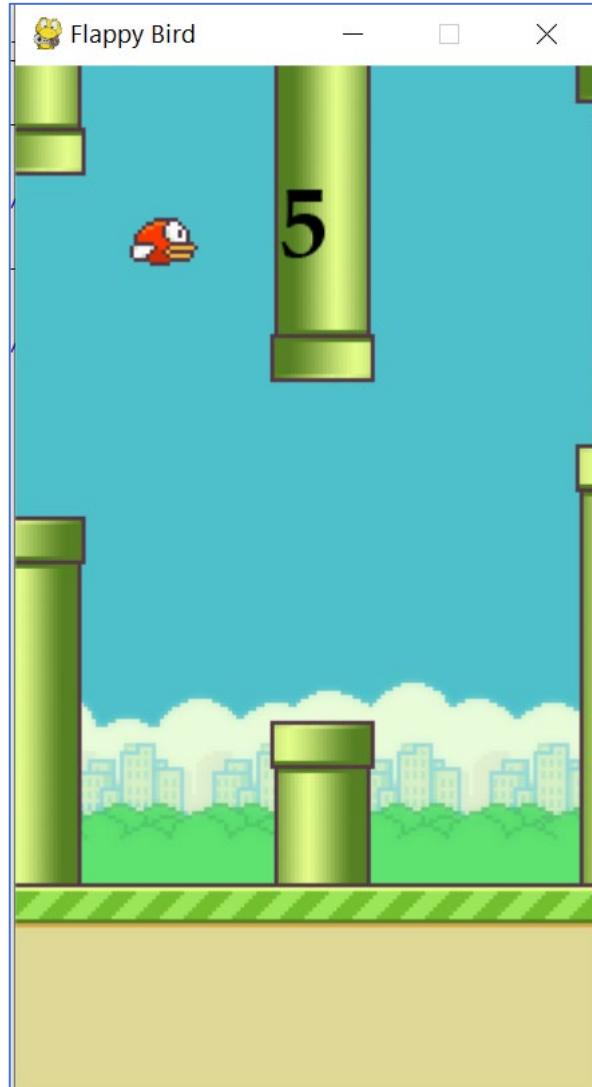
با نمایش متن، بازی به روز و سپس متوقف می شود تا زمانی که بازیکن با فراخوانی تابع `waitForPlayerToPressKey()`، کلیدی را فشار دهد.

با فشار دادن یک کلید، افکت صوتی game over ممکن است همچنان وجود داشته باشد یا نباشد. برای توقف این جلوه صوتی قبل از شروع بازی جدید، کد زیر را می نویسیم:

```
gameOverSound.stop()
```

THE END!

پروژه نهایی



بازی معروف : flappy

در این بازی پرنده دارای حرکت دائمی رو به پایین است. با استفاده از کلیدهای جهت نما پرنده را به بالا هدایت می کنیم تا به لوله ها برخورد نکند. هنگامی که پرنده از بین دو لوله عبور می کند، یک امتیاز به بازیکن اضافه می شود. باخت بازیکن با برخورد به لوله ها اتفاق می افتد.

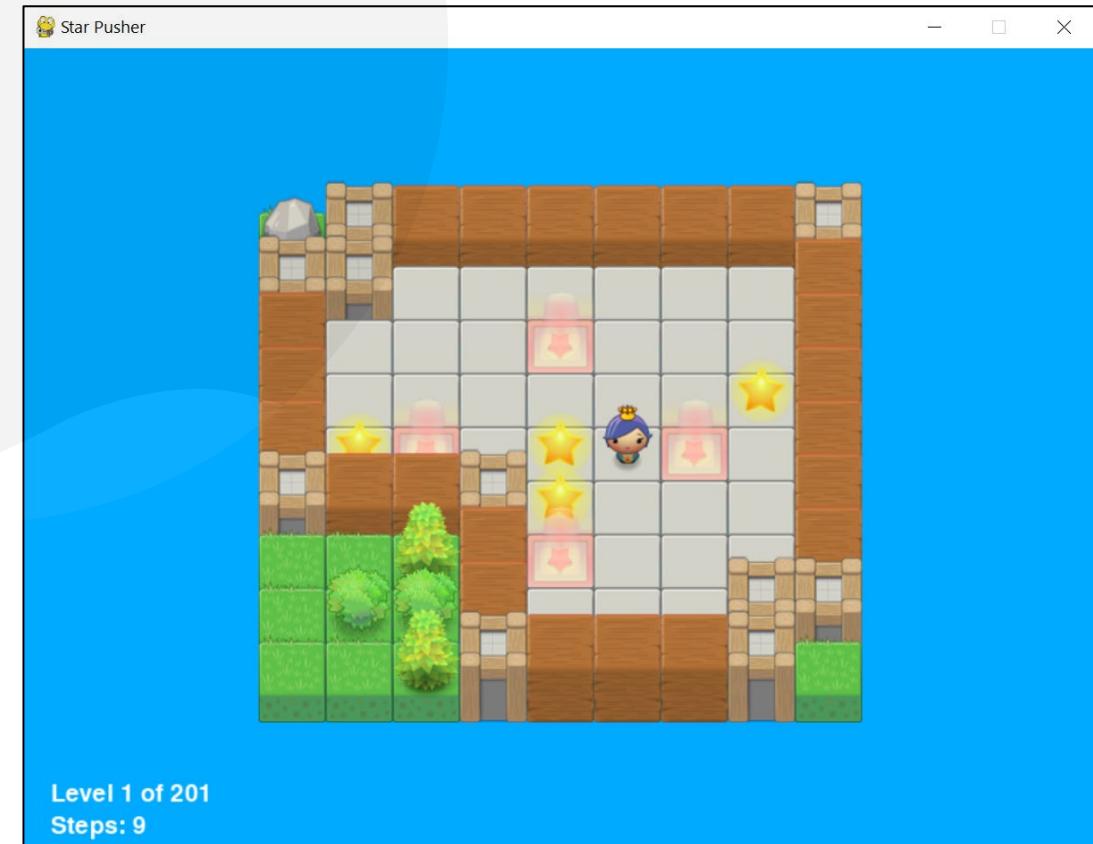
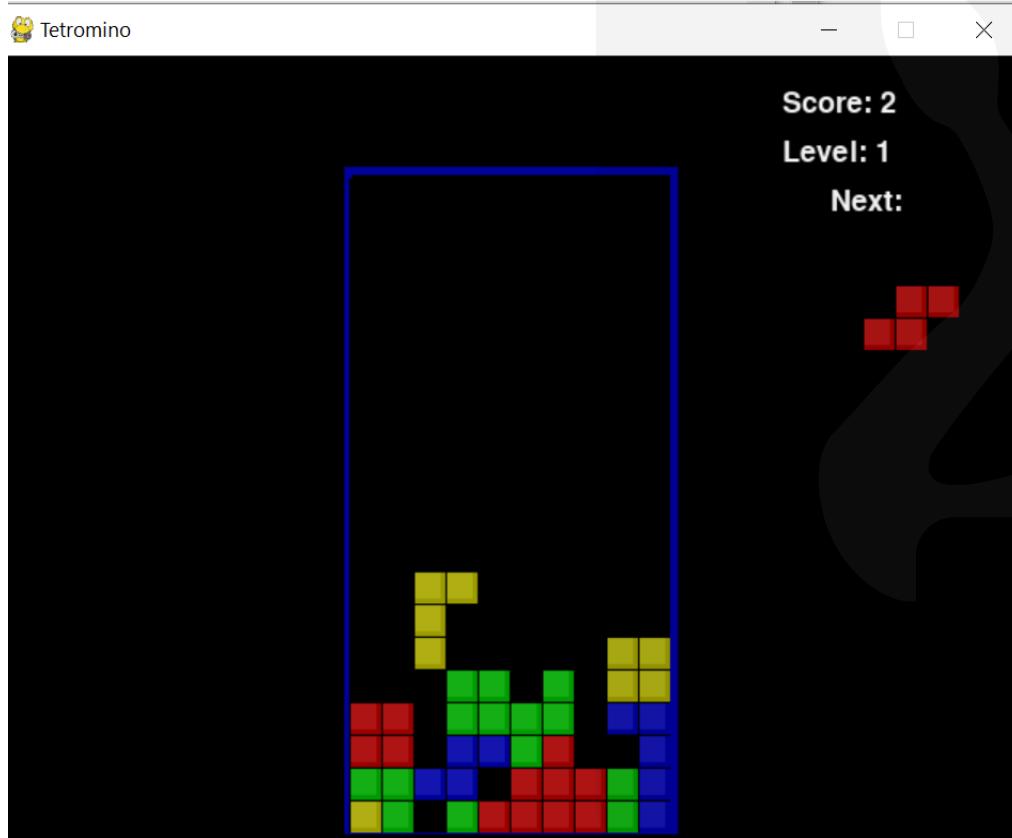
همه فایلهای مربوط به این قسمت در سایت بارگذاری شده است:

<http://pythontreek.com/files/pygame>

بازی های بیشتر

کد بازیهای زیر در سایت موجود است:

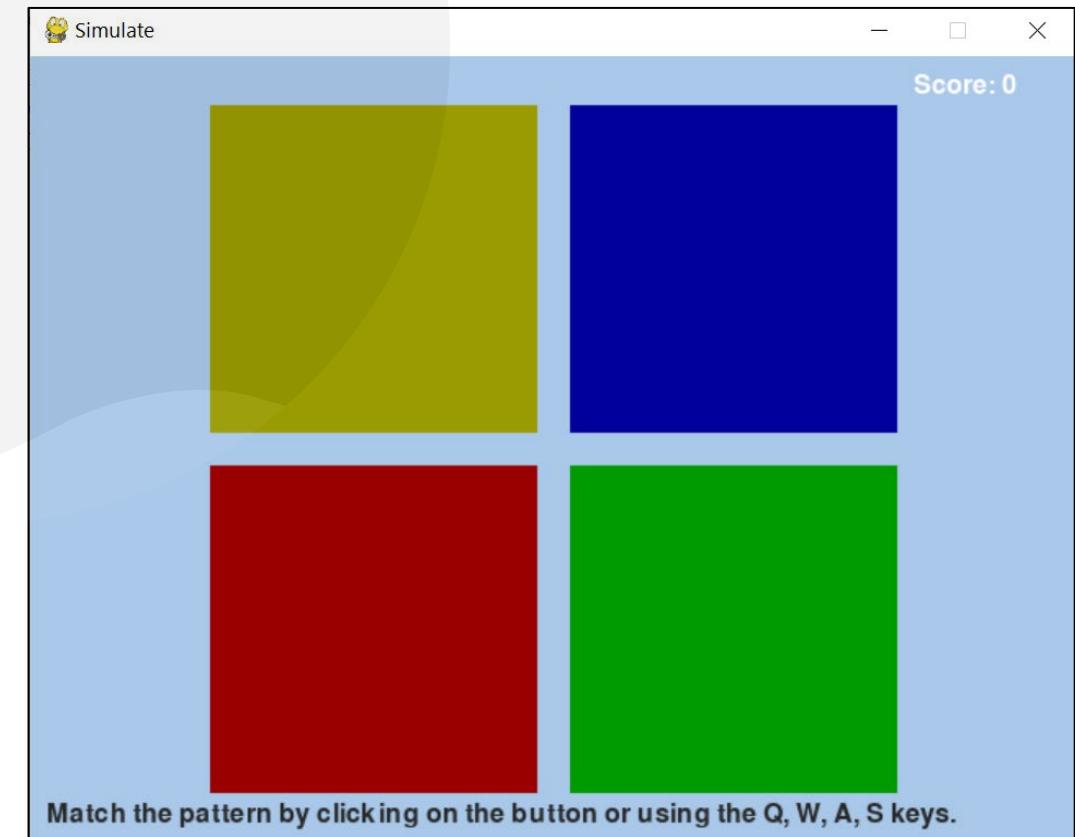
<http://pythonteeek.com/files/fmp>



بازی های بیشتر

کد بازیهای زیر در سایت موجود است:

<http://pythonteeek.com/files/fmp>



بازی های بیشتر

کد بازیهای زیر در سایت موجود است:

<http://pythonteeek.com/files/fmp>



15

