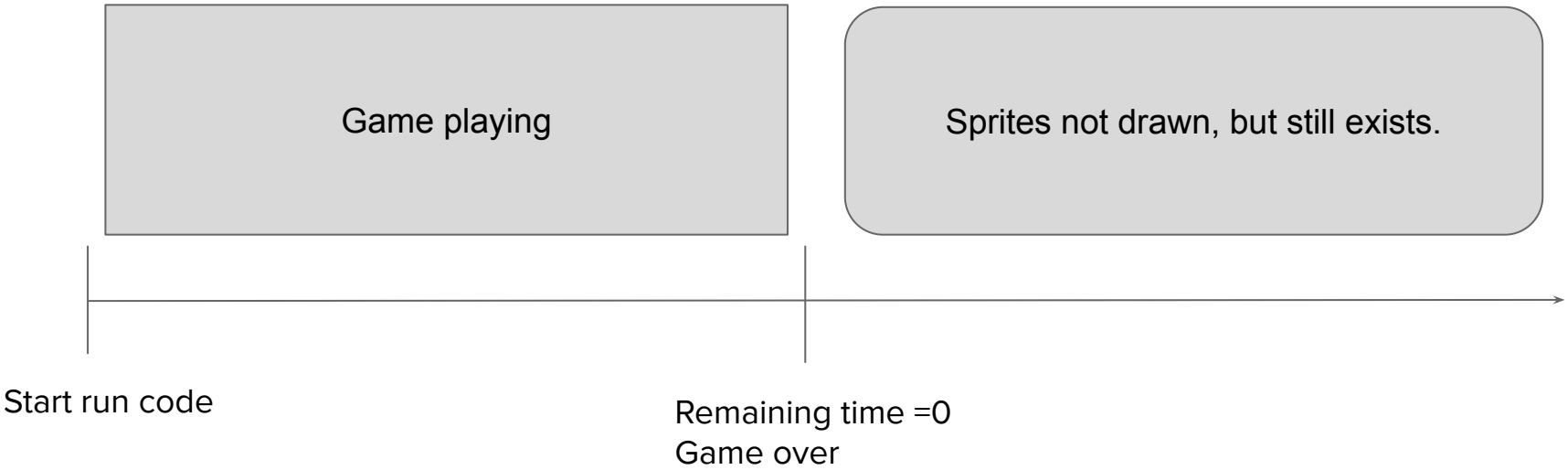
Game

part7



Hmmmm... Keep clicking on the game over screen

THINK?

When “game over” show up, we can still click squares even though we can not see them.

How to fix it?

Remove all sprites in group...

Empty all sprites when game over

```
import pygame
...
class Square(pygame.sprite.Sprite):
    def __init__(self, x, y, side, speed_x, speed_y, colour):
        super().__init__()
        self.side = side
        self.image = pygame.Surface([side, side])

        self.speed_x=0
        self.speed_y=0
...
start_time = pygame.time.get_ticks()
time_left = 3000
clicked_count = 0

done = False
while not done:
...
pygame.quit()
```

For our convenience, let's make sure all the Square do not move temporary. So inside the constructor of Square, we makes its speed to zero. You can still keep the randomization code, but that the speed from that will not be used.

And reduce the time for faster result.

Empty all sprites when game over

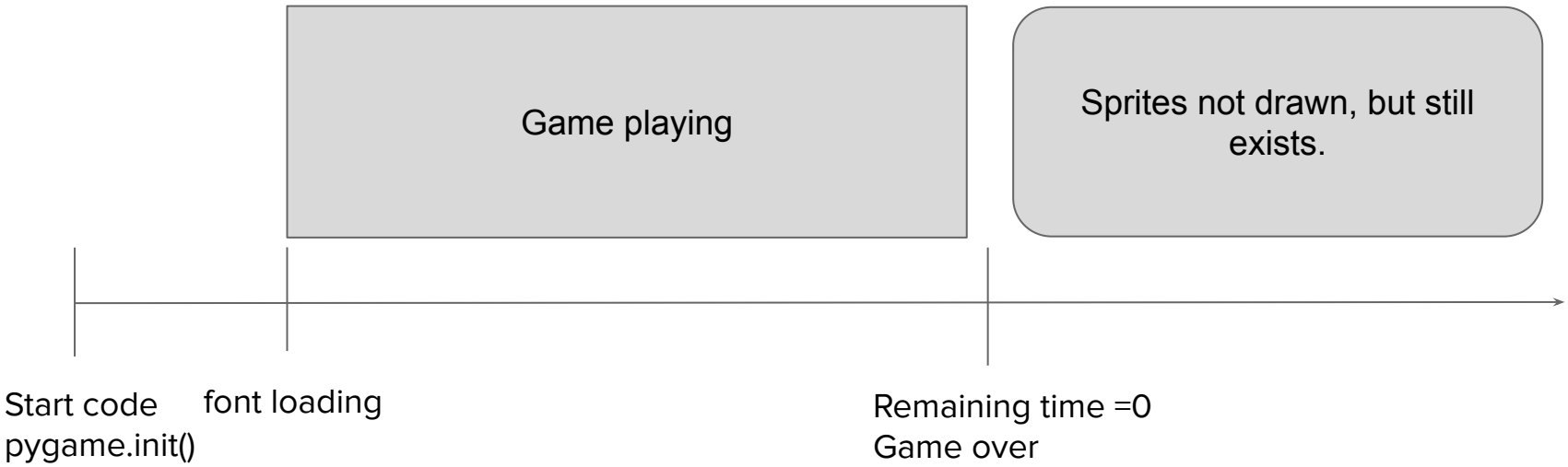
```
import pygame
...
remaining_time = (time_left-pygame.time.get_ticks()+start_time+500)//1000
    if remaining_time <= 0:
        remaining_time = 0
        over_text = font.render("Game Over", False, (0, 255, 0), (0, 0, 255))
        screen.blit(over_text, (200,300))
        allspriteslist.empty()

    else:
        allspriteslist.draw(screen)
...

    clock.tick(120)
pygame.quit()
```

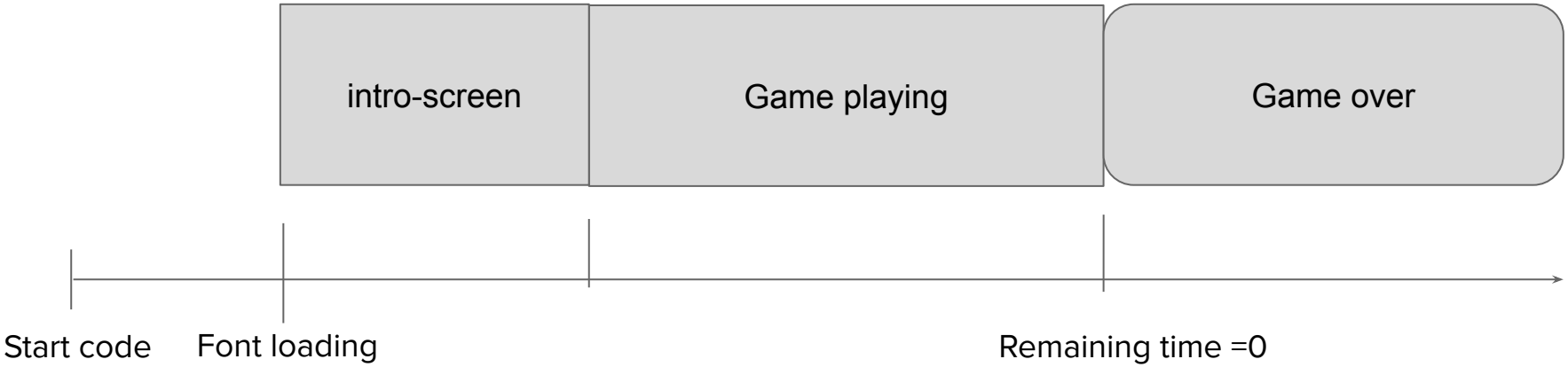
There are at least two methods to fix:
1, selectively call the update
2, empty the list
3, stopped updating clicked count

And we do it the hard way.


Game playing
Sprites not drawn, but still exists.
Start code
pygame.init()
font loading
Remaining time =0
Game over

intro-screen

What is intro-screen?



Intro-screen ---- new boolean variable

```
import pygame
...
start_time = pygame.time.get_ticks()
time_left = 13000
clicked_count = 0

showIntro = True

done = False
while not done:

...
pygame.quit()
```

Having an intro screen meaning that we have one more stage to deal with. So we are adding one more variable to keep track of our game status.

Intro-screen ---- new boolean variable

```
import pygame
...
showIntro = True

done = False
while not done:
    remaining_time = (time_left-pygame.time.get_ticks()+start_time+500)//1000

    if showIntro is True:
        print(“intro-screen”)
    if remaining_time <= 0:
        remaining_time = 0
        over_text = font.render("Game Over", False, (0, 255, 0), (0, 0, 255))
        screen.blit(over_text, (200,300))
        allspriteslist.empty()
    else:
        allspriteslist.draw(screen)
...
pygame.quit()
```

Let’s quickly print some text to make sure it is working.

Intro-screen ---- new boolean variable

```
import pygame
...
showIntro = True

done = False
while not done:
    remaining_time = (time_left-pygame.time.get_ticks()+start_time+500)//1000
    if showIntro is True:
        print("intro-screen")
        intro_text = font.render("Welcome", False, (0, 255, 0), (0, 0, 255))
        screen.blit(intro_text, (100, 100))

    if remaining_time <= 0:
        remaining_time = 0
        over_text = font.render("Game Over", False, (0, 255, 0), (0, 0, 255))
        screen.blit(over_text, (200,300))
        allspriteslist.empty()
    else:
        allspriteslist.draw(screen)
...
pygame.quit()
```

Once we see it is working, it is time to display text in the game instead of displaying it in the console.

Intro-screen ---- new boolean variable

```
import pygame
...
showIntro = True

done = False
while not done:
    remaining_time = (time_left-pygame.time.get_ticks()+start_time+500)//1000

    if showIntro is True: #intro-screen
        intro_text = font.render("Welcome", False, (0, 255, 0), (0, 0, 255))
        screen.blit(intro_text, (100, 100))
    if remaining_time <= 0: #game over
        remaining_time = 0
        over_text = font.render("Game Over", False, (0, 255, 0), (0, 0, 255))
        screen.blit(over_text, (200,300))
        allspriteslist.empty()
    else: #game playing
        allspriteslist.draw(screen)
...
pygame.quit()
```

We could put some comment in our code to better represent what each chunk of code does. Anything comes after the # key is ignored by python.

It means they are not considered as programming code.

Intro-screen ---- new boolean variable

```
import pygame
...
showIntro = True

done = False
while not done:
    remaining_time = (time_left-pygame.time.get_ticks()+start_time+500)//1000

    if showIntro is True: #intro-screen
        intro_text = font.render("Welcome", False, (0, 255, 0), (0, 0, 255))
        screen.blit(intro_text, (100, 100))
    if remaining_time <= 0: #game over
        remaining_time = 0
        over_text = font.render("Game Over", False, (0, 255, 0), (0, 0, 255))
        screen.blit(over_text, (200,300))
        allspriteslist.empty()
    else: #game playing
        allspriteslist.draw(screen)
...
pygame.quit()
```

If you run your code now, you might notice something weird.
Let's recap a little bit here.

```
if blah blah:
   # result 1
elif blah blah blah:
   # result 2
elif blah blah blah:
   # result 3
else:
   # result 4
```

```
if blah blah:
   # result 1
elif blah blah blah:
   # result 2
```

```
if blah blah:
   # result 1
else:
   # result 2
```

```
if blah blah:
   # result 1
```

This is a complete if-else block

But as you can see, elif and else are optional. You only use it when you want the other side of the world.

Make sure your understand this. In all these example here, there is only one result each time it executes. We can not have more than one result at a time.

```
if blah blah:
   # result 1
else:
   # result 2

if blah blah blah:
   # result 3
```

What about this? How many possibilities are there?

```
if blah blah:
   # result A
else:
   # result B

if blah blah blah:
   # result C
```

This part will give
A or B.
NOT A and B.
It will give you
only one result

This part will give
you C or NOTHING.

So in the end, the results are:
AC
A
BC
B

```
if blah blah:
   # result A
elif blah blah:
   # result B


if blah blah blah:
   # result D
else:
   # result E


if blah blah blah:
   # result F
```

As you can see, I have the intention to group them into their group.

What are the possibilities?

```
if blah blah:
   # result A
elif blah blah:
   # result B

if blah blah blah:
   # result D
else:
   # result E

if blah blah blah:
   # result F
```

ADF
AEF
AD
AE

BDF
BEF
BD
BE

DF
D

EF
E

Can there be DE?
Can there be AB?
Can there be NONE?

Intro-screen ---- change to next step (event)

```
import pygame
...
done = False
while not done:
    for event in pygame.event.get():
        if event.type == pygame.MOUSEBUTTONDOWN:
            pos = pygame.mouse.get_pos()
            for sprite in allspriteslist:
                if sprite.rect.collidepoint(pos):
                    clicked_count = clicked_count + 1
                    sprite.remove(allspriteslist)
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_q:
                done = True
            if event.key == pygame.K_b:
                background_colour = (0,0,200)
            if event.key == pygame.K_g:
                background_colour = (0,200,0)
            if event.key == pygame.K_SPACE:
                showIntro = False
...
pygame.quit()
```

Here is the key point. showIntro could remain the same forever and we only see the intro-screen. That is what the while loop is doing. Inside the while loop our logic is trying to decide what to show. Intro? Gameplay? Gameover?

An event would change the whole story. It is not time-driven, but event-driven

What happens?
How to fix it?

Intro-screen ---- complete if statement

```
import pygame
...
showIntro = True

done = False
while not done:
    remaining_time = (time_left-pygame.time.get_ticks()+start_time+500)//1000

    if showIntro is True: #intro-screen
        intro_text = font.render("Welcome", False, (0, 255, 0), (0, 0, 255))
        screen.blit(intro_text, (100, 100))
    elif remaining_time <= 0: #game over
        remaining_time = 0
        over_text = font.render("Game Over", False, (0, 255, 0), (0, 0, 255))
        screen.blit(over_text, (200,300))
        allspriteslist.empty()
    else: #game playing
        allspriteslist.draw(screen)
...
pygame.quit()
```

The problem is similar. It is flow-control. We want only one result, or one out of three result.
So if intro screen is showing, no need to show others.

Intro-screen ---- complete if statement

```
import pygame
...
done = False
while not done:
    remaining_time = (time_left-pygame.time.get_ticks()+start_time+500)//1000
    if showIntro is True: #intro-screen
        intro_text = font.render("Welcome", False, (0, 255, 0), (0, 0, 255))
        screen.blit(intro_text, (100, 100))
    elif remaining_time <= 0: #game over
        remaining_time = 0
        over_text = font.render("Game Over", False, (0, 255, 0), (0, 0, 255))
        screen.blit(over_text, (200,300))
        allspriteslist.empty()
    else: #game playing
        allspriteslist.draw(screen)
    time = font.render(str(remaining_time), False, (0, 255, 0), (0, 0, 255))
    screen.blit(time, (100, 100))
    clicked_count_text = font.render(str(clicked_count), False, (0, 255, 0), (0, 0, 255))
    screen.blit(clicked_count_text, (300, 100))
...
pygame.quit()
```

The part of the code is display time and clicked count. It makes more sense to display them during the game. Of course some might think it is better to show clicked count on the gameover screen. I will leave that to you.

Intro-screen ---- complete if statement

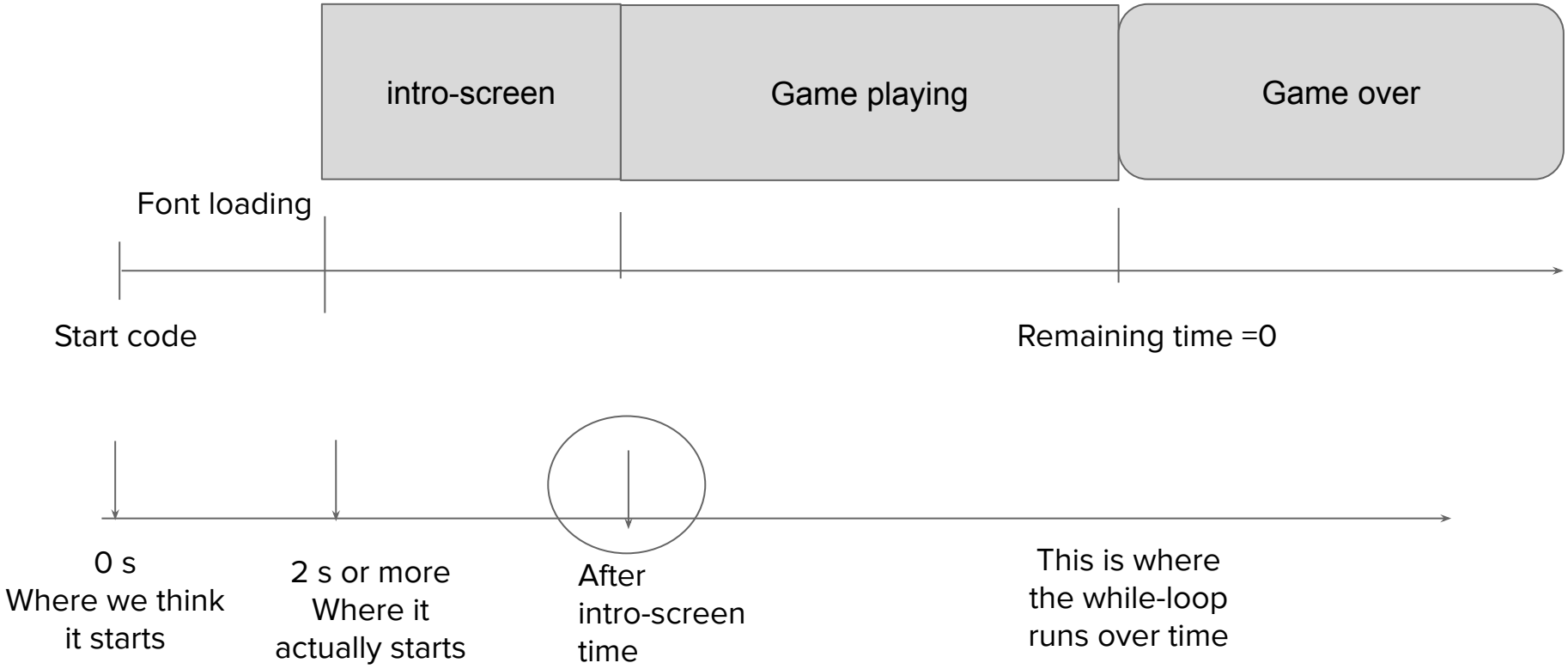
```
import pygame
...
done = False
while not done:
    remaining_time = (time_left-pygame.time.get_ticks()+start_time+500)//1000
    if showIntro is True: #intro-screen
        intro_text = font.render("Welcome", False, (0, 255, 0), (0, 0, 255))
        screen.blit(intro_text, (100, 100))
    elif remaining_time <= 0: #game over
        remaining_time = 0
        over_text = font.render("Game Over", False, (0, 255, 0), (0, 0, 255))
        screen.blit(over_text, (200,300))
        allspriteslist.empty()
    else: #game playing
        allspriteslist.draw(screen)
        time = font.render(str(remaining_time), False, (0, 255, 0), (0, 0, 255))
        screen.blit(time, (100, 100))
        clicked_count_text = font.render(str(clicked_count), False, (0, 255, 0), (0, 0, 255))
        screen.blit(clicked_count_text, (300, 100))
...
pygame.quit()
```

We can also move the code that displays time and clicked count under the stage where game is going on.

If you see game-over screen after pressing space, that means gameplay has been skipped. We should give it more time for now.

```
time_left = 10000
```

More TIME


intro-screen
Game playing
Game over
Font loading
Start code
Remaining time =0
0 s
Where we think
it starts
2 s or more
Where it
actually starts
After
intro-screen
time
This is where
the while-loop
runs over time

More TIME ---- change start-time

```
import pygame
...
font = pygame.font.SysFont("comicsansms", 72)

start_time = pygame.time.get_ticks()
# or start_time = 0

time_left = 3000
clicked_count = 0
showIntro = True

done = False
while not done:
...
    for event in pygame.event.get():

        if event.key == pygame.K_SPACE:
            showIntro = False
            start_time = pygame.time.get_ticks()
...
pygame.quit()
```

Keep both

First we will need to create a variable start_time the value here does matter, it will be overwritten soon.

We need to get the latest time as soon as the game starts. And this is where we receive the space key event.

What happens when you press SPACE more than once?
How to fix it?

More TIME ---- change start-time

```
import pygame
...
font = pygame.font.SysFont("comicsansms", 72)

start_time = pygame.time.get_ticks()

time_left = 3000
clicked_count = 0
showIntro = True

done = False
while not done:
...
    for event in pygame.event.get():

            if event.key == pygame.K_SPACE and showIntro is True:
                showIntro = False
                start_time = pygame.time.get_ticks()


...
pygame.quit()
```

All we need to do is to more sure the game only can use the space key once. To do that, we add a little more on the if statement to make sure after showIntro is changed to FALSE, we won't be able to enter the code inside this if block

Change speed by different sizes

```
import pygame
...
class Square(pygame.sprite.Sprite):
    def __init__(self, x, y, side, speed_x, speed_y, colour):
        super().__init__()
        self.side = side
        self.image = pygame.Surface([side, side])

        self.speed_x=speed_x
        self.speed_y=speed_y
...
start_time = pygame.time.get_ticks()
time_left = 3000
clicked_count = 0

done = False
while not done:
...
pygame.quit()
```

Now we can change back to the more flexible version.

Let's try to make the big moves after than the smaller one.

Change speed by different sizes

```
import pygame
...
for i in range(60):
    sc = random.randint(30, 255)
    x_pos = random.randint(0, 800)
    y_pos = random.randint(0, 600)
    size = random.randint(10, 20)

    x_speed = random.randint(-1, 1) * size/5.0
    y_speed = random.randint(-1, 1) * size/5.0

    s = Square(x_pos, y_pos, size, x_speed, y_speed, (sc, sc, sc))
    allspriteslist.add(s)

pygame.init() # 0 s

...
pygame.quit()
```

Do you see a problem here?

Change speed by different sizes

```
import pygame
...
for i in range(60):
    sc = random.randint(30, 255)
    x_pos = random.randint(0, 800)
    y_pos = random.randint(0, 600)
    size = random.randint(10, 20)

    x = 0
    while x == 0:
        x = random.randint( -1, 1 )
    y = 0
    while y == 0:
        x = random.randint( -1, 1 )

    x_speed = x * size/5
    y_speed = y * size/5

    s = Square(x_pos, y_pos, size, x_speed, y_speed, (sc, sc, sc))
    allspriteslist.add(s)
pygame.init() # 0 s
```

This is not the best, try this.

challenge

Change speed according to color