Game 2

part1

Surface

```
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

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour
        self.image.fill(colour)
```

What is Surface?

Surface is a class. How do we get to know more about it?

Surface

```
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

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour
        self.image.fill(colour)
```

It is an image, as the appearance of the object. Remember, it is just the image.

So we named it image

Surface

```
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

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour
        self.image.fill(colour)
```

Create a Surface **object** in pygame file, named by **image**.

Think about this Surface is a artboard/paper that we are about to draw on.

Surface

```
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

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour
        self.image.fill(colour)
```

What we did was to create a drawing space of as big as side by side. And we simply filled the whole space with one colour.

Fill colour into surface

We never said how much we want to fill, but it just did. This also means the limit is the size we gave.

Surface

Screen is also surface. Remember how we draw it?

allspritelist.draw(screen)

```
pygame.init() # 0 s

screen = pygame.display.set_mode([800,600])
pygame.display.set_caption('Snake Example')
clock = pygame.time.Clock()

background_colour = (0,0,0)


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

start_time = pygame.time.get_ticks()
time_left = 10000
clicked_count = 0

showIntro = True

done = False
while not done:
```

rect

```
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

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour
        self.image.fill(colour)
```

The image was all for the appearance. Based on the Surface, we can get a Rectangle object. And the size is exactly the same as the Surface.

rect

```
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

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour
        self.image.fill(colour)
```

Rect means **rectangle**

```
get_rect()
```

get the rectangular area of the Surface

```
get_rect(**kwargs) -> Rect
```

rect

```
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

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour
        self.image.fill(colour)
```

- Return a new rectangle covering the entire surface
- Manipulation
  - Position
  - Interaction

rect

```
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

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour
        self.image.fill(colour)
```

All these have nothing to do with its appearance, but only location, orientation, and interaction.

Draw Circle

```
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

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y

        self.colour = colour
        self.image.fill(colour)
...
pygame.quit()
```

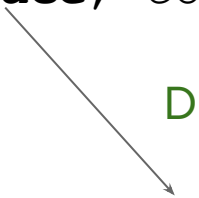
We can try modify the Square and make it draw a circle. First let's make them all stop.

Draw Circle

```
pygame.draw.circle()
    draw a circle
    circle(surface, color, center, radius) -> Rect
    circle(surface, color, center, radius, width=0,
    draw_top_right=None, draw_top_left=None, draw_bottom_left=None,
    draw_bottom_right=None) -> Rect
```

Draws a circle on the given surface.

`circle(`**`surface`**`, color, center, radius)`



Draw a circle on the given surface

pygame.draw.circle(**screen**, color, center, radius)

Draw Circle

```
pygame.draw.circle()
    draw a circle
    circle(surface, color, center, radius) -> Rect
    circle(surface, color, center, radius, width=0,
    draw_top_right=None, draw_top_left=None, draw_bottom_left=None,
    draw_bottom_right=None) -> Rect
```

Draws a circle on the given surface.

```
circle(surface, color, center, radius)
```

Color of the circle

pygame.draw.circle(screen, **(200,0,0)**, center, radius)

Draw Circle

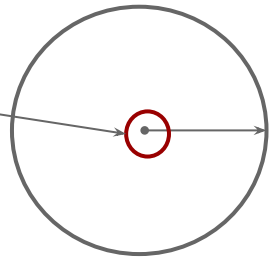
```
pygame.draw.circle()
    draw a circle
    circle(surface, color, center, radius) -> Rect
    circle(surface, color, center, radius, width=0,
    draw_top_right=None, draw_top_left=None, draw_bottom_left=None,
    draw_bottom_right=None) -> Rect
```

Draws a circle on the given surface.

`circle(surface, color, center, radius)`

Center point of the circle

pygame.draw.circle(screen, (200,0,0), **(400, 300)**, radius)

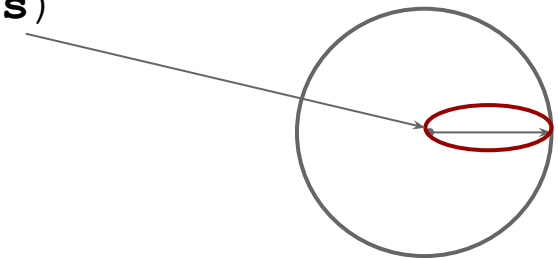


Draw Circle

```
pygame.draw.circle()
    draw a circle
    circle(surface, color, center, radius) -> Rect
    circle(surface, color, center, radius, width=0,
    draw_top_right=None, draw_top_left=None, draw_bottom_left=None,
    draw_bottom_right=None) -> Rect
```

Draws a circle on the given surface.

`circle(surface, color, center, radius)`



pygame.draw.circle(screen, (200,0,0), (400, 300), **25**)

Draw Circle

pygame.draw.circle(screen, (200,0,0), (400, 300), 25)

Tuple Review:

What's the difference between tuple and list ?

Draw Circle

pygame.draw.circle(screen, (200,0,0), (400, 300), 25)

List and Tuple are really similar, but you cannot change the value after you create the Tuple.

List
```
a = [0,3,5]
print(a[1])
a[1] = 10
```

Tuple
```
a = (0,3,5)
print(a[1])
a[1] = 10     # This is wrong.
```

Draw Circle

This is drawing a circle only. It has no physical body to interact with.

```
import pygame
...

done = False
while not done:
...

    screen.fill(background_colour)
    pygame.draw.circle(screen, (200,0,0), (400, 300), 25)

    remaining_time = (time_left-pygame.time.get_ticks()+start_time+500)//1000

...
pygame.quit()
```

Draw Circle in class Square

```
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

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour

        pygame.draw.circle(screen, (200,0,0), (400, 300), 25)

        self.image.fill(colour)
...
pygame.quit()
```

What if we move this line into the Class?

Get ERROR!

```
============== RESTART: /Users/eavanwang/Desktop/week27start.py ==============
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
Traceback (most recent call last):
  File "/Users/eavanwang/Desktop/week27start.py", line 52, in <module>
    s = Square(x_pos, y_pos, size, x_speed, y_speed, (sc, sc, sc))
  File "/Users/eavanwang/Desktop/week27start.py", line 18, in __init__
    pygame.draw.circle(screen, (200,0,0), (400, 300), 25)
NameError: name 'screen' is not defined
>>>
```

How to fix it?
Before we fix it, we need to know what the problem is.
There are two problems.

Draw Circle in class Square

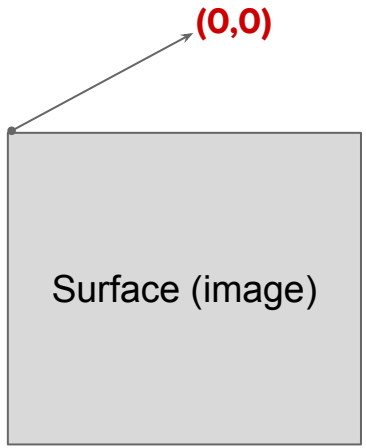
```
import pygame
...
class Square(pygame.sprite.Sprite):
    def __init__(self, x, y, side, speed_x, speed_y, colour):
        super().__init__()
        self.side = side
        self.image = pygame.Surface([side, side])
        self.speed_x=0
        self.speed_y=0

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour

        pygame.draw.circle(self.image, (200,0,0), (400, 300), 25)

...
pygame.quit()
```

We need to draw a circle on the Surface on for each Object. Which Surface?

What happens?
Why?

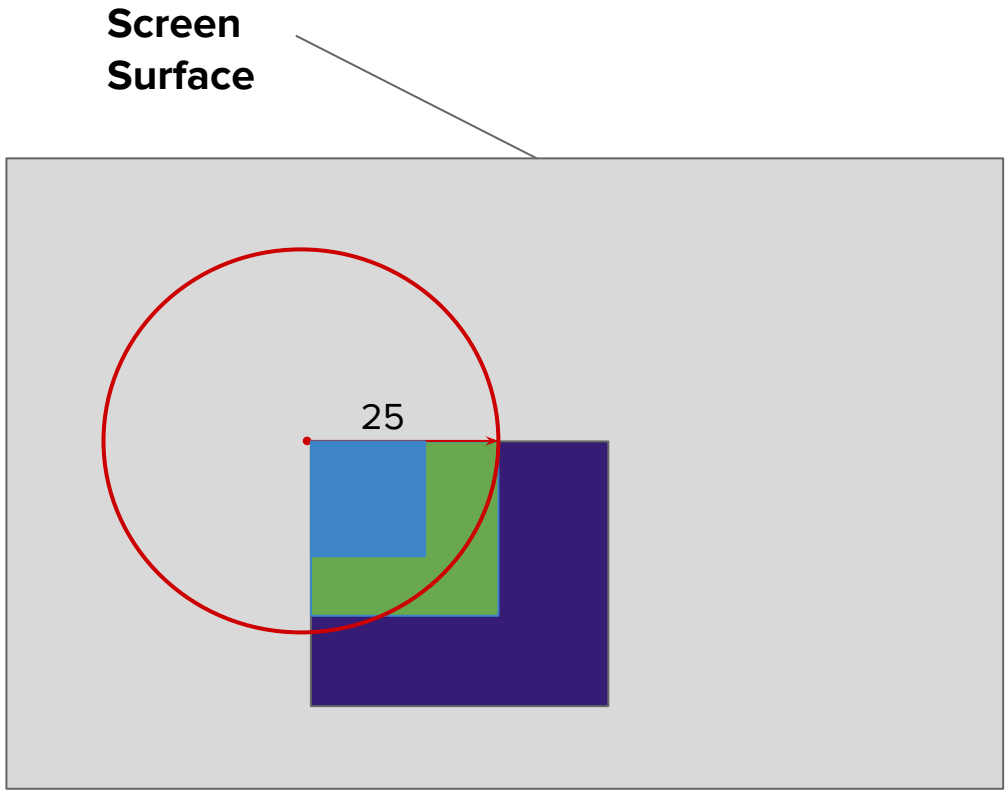
Draw Circle in class Square

```
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

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour

        pygame.draw.circle(self.image, (200,0,0), (0,0), 25)

...
pygame.quit()
```



Where exactly on the Surface?

Observe circles displayed

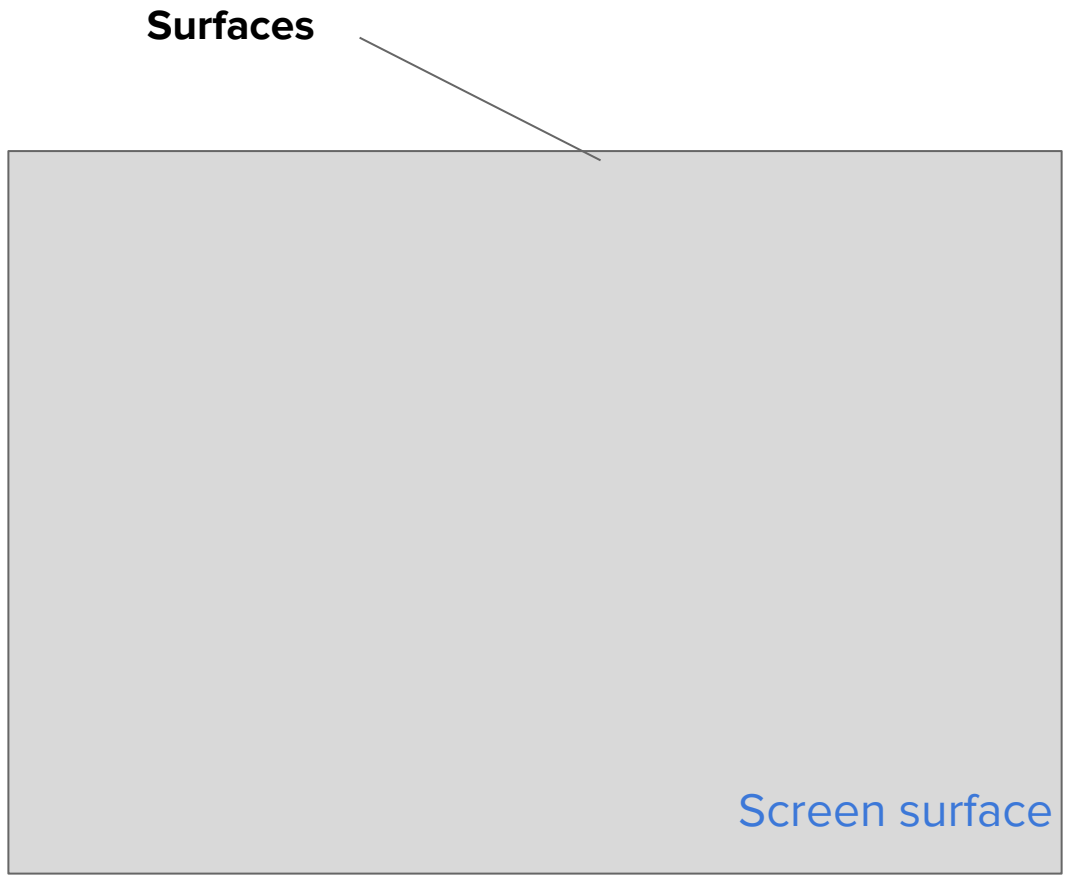


Draw Circle in class Square

```
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

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour

        pygame.draw.circle(self.image, (200,0,0), (0,0), side//2)

...
pygame.quit()
```

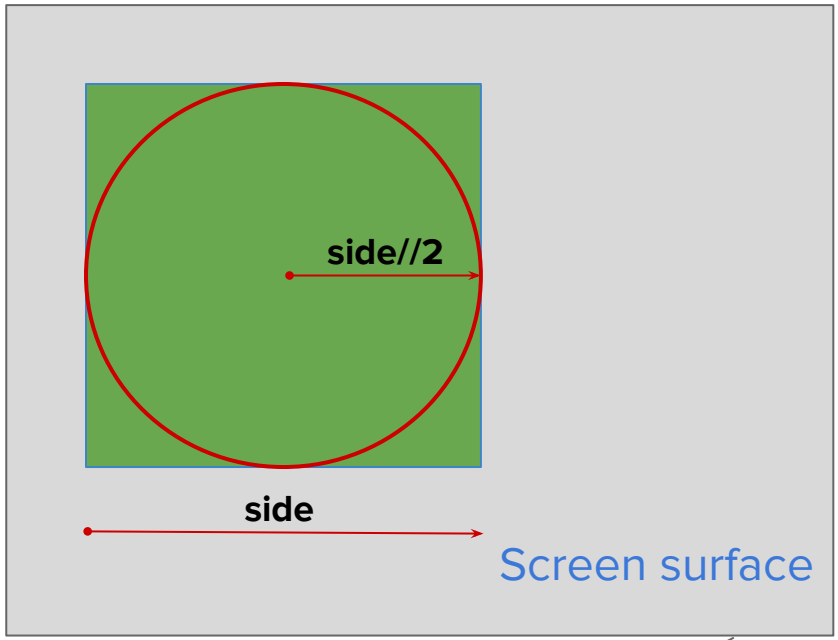
How big is the circle?

Observe circles displayed



Draw Circle in class Square

```
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

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour

        pygame.draw.circle(self.image, (200,0,0), (side//2, side//2), side//2)

...
pygame.quit()
```



Self.rect only matters on the position of the whole Object

Way 2 to Draw Circle in class Square

```
pygame.gfxdraw.filled_circle()
```

draw a filled circle

```
filled_circle(surface, x, y, r, color) -> None
```

Draws a filled circle on the given surface. For an unfilled circle use `circle()`.

- **surface** (***Surface***) -- surface to draw on
- **x** (*int*) -- x coordinate of the center of the circle
- **y** (*int*) -- y coordinate of the center of the circle
- **r** (*int*) -- radius of the circle
- **color** -- color to draw with, the alpha value is optional if using a tuple `(RGB[A])`

Way 2 to Draw Circle in class Square

`pygame.gfxdraw.filled_circle()`

draw a filled circle

`filled_circle(surface, x, y, r, color) -> None`

Draws a filled circle on the given surface. For an unfilled circle use `circle()`.

```
filled_circle(surface, x, y, r, color)
```

pygame.gfxdraw.filled_circle(self.image, side//2, side//2, side//2-1, (200,0,0))

Way 2 to Draw Circle in class Square

```
import pygame
import random
import pygame.gfxdraw

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

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour

        pygame.gfxdraw.filled_circle(self.image, side//2, side//2, side//2, (200,0,0))
```

Way 2 to Draw Circle in class Square

```
import pygame
import random
import pygame.gfxdraw

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], pygame.SRCALPHA)

        self.speed_x=0
        self.speed_y=0

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour

        pygame.gfxdraw.filled_circle(self.image, side//2, side//2, side//2, (200,0,0))
```

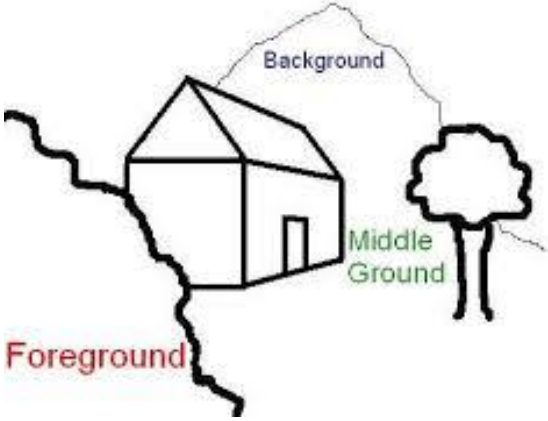
Observe circles displayed

Way 2 to Draw Circle in class Square

```
import pygame
import random
import pygame.gfxdraw

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], pygame.SRCALPHA)

        self.speed_x=0
        self.speed_y=0

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour

        pygame.gfxdraw.filled_circle(self.image, side//2, side//2, side//2-1, (200,0,0))
```

What is parallax ?

It has something to do with foreground, midground and background.



parallax

```
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], pygame.SRCALPHA)

        self.speed_x=speed_x
        self.speed_y=0

        self.rect=self.image.get_rect()
        self.rect.x=x
        self.rect.y=y
        self.colour = colour

        pygame.gfxdraw.filled_circle(self.image, side//2, side//2, side//2-1, (200,0,0))
...
pygame.quit()
```

Let's try to make them with only speed_x

parallax

```
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], pygame.SRCALPHA)

        self.speed_x=speed_x
        self.speed_y=0
...
for i in range(60):
...
    x = 0
    while x == 0:
        x = random.randint(0, 1)
    y = 0
    while y == 0:
        y = random.randint(-1, 1)
...
pygame.quit()
```

And only positive speeds, so that they all move to the right in different speed.

Bounce back

```
import pygame
...
class Square(pygame.sprite.Sprite):
    def __init__(self, x, y, side, speed_x, speed_y, colour):
...
    def update(self):

        self.rect.x = self.rect.x + self.speed_x
        self.rect.y = self.rect.y + self.speed_y

        if self.rect.x > 800:
            self.rect.x = -self.side
            self.speed_x = -self.speed_x

        if self.rect.x < -self.side:
            self.rect.x = 800
        if self.rect.y > 600:
            self.rect.y = -self.side
        if self.rect.y < -self.side:
            self.rect.y = 600
...
pygame.quit()
```

Since it is a ball, it should be BOUNCY!

Instead of relocating the object when it hits the right side, we will change its speed to negative.

Observe...

They bounce too late!

Bounce back

```
import pygame
...
class Square(pygame.sprite.Sprite):
    def __init__(self, x, y, side, speed_x, speed_y, colour):
...
    def update(self):

        self.rect.x = self.rect.x + self.speed_x
        self.rect.y = self.rect.y + self.speed_y

        if self.rect.x > 800 - self.side:
            self.speed_x = -self.speed_x

        if self.rect.x < -self.side:
            self.rect.x = 800
        if self.rect.y > 600:
            self.rect.y = -self.side
        if self.rect.y < -self.side:
            self.rect.y = 600
...
pygame.quit()
```

We are still using the top left corner of the rect for detection. It shouldn't need to reach 800 before the ball bounces.

Observe...

It Flickers!

Bounce back

```
import pygame
...
class Square(pygame.sprite.Sprite):
    def __init__(self, x, y, side, speed_x, speed_y, colour):
...
    def update(self):

        self.rect.x = self.rect.x + self.speed_x
        self.rect.y = self.rect.y + self.speed_y

        if self.rect.x >  800 - self.side:
            self.rect.x = 800 - self.side
            self.speed_x = -self.speed_x

        if self.rect.x < -self.side:
            self.rect.x = 800
        if self.rect.y > 600:
            self.rect.y = -self.side
        if self.rect.y < -self.side:
            self.rect.y = 600
...
pygame.quit()
```

We can force it to be at the mostright position it should be. And let the speed do its job.

In the next frame, it should already be moved with the negative speed.

One more thing. >= is better than >

QUIT event

What is QUIT ?

```
import pygame
...
done = False
while not done:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:
            print(quit)

        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)

...
pygame.quit()
```

This is an event. It is trigger when you click the close button in your game.

QUIT event

```
import pygame
...
done = False
while not done:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:
            print(quit)
            pygame.quit()

        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)

...
pygame.quit()
```

QUIT event

```
import pygame
...
done = False
while not done:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:
            pygame.quit()
            exit()

        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)

...
pygame.quit()
```

A bonus to clean up memory on your computer