Game2

Part 3

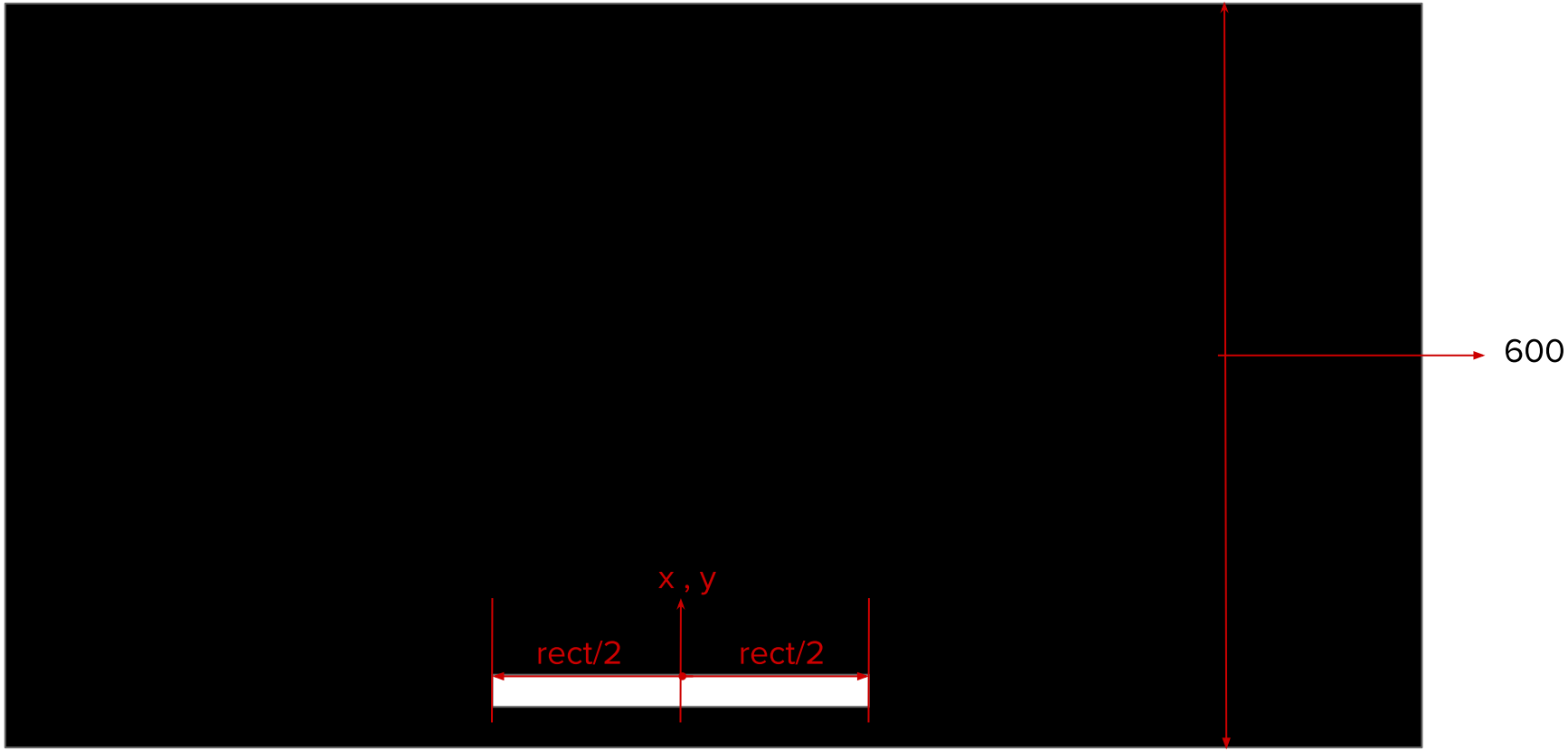
Observe your game...

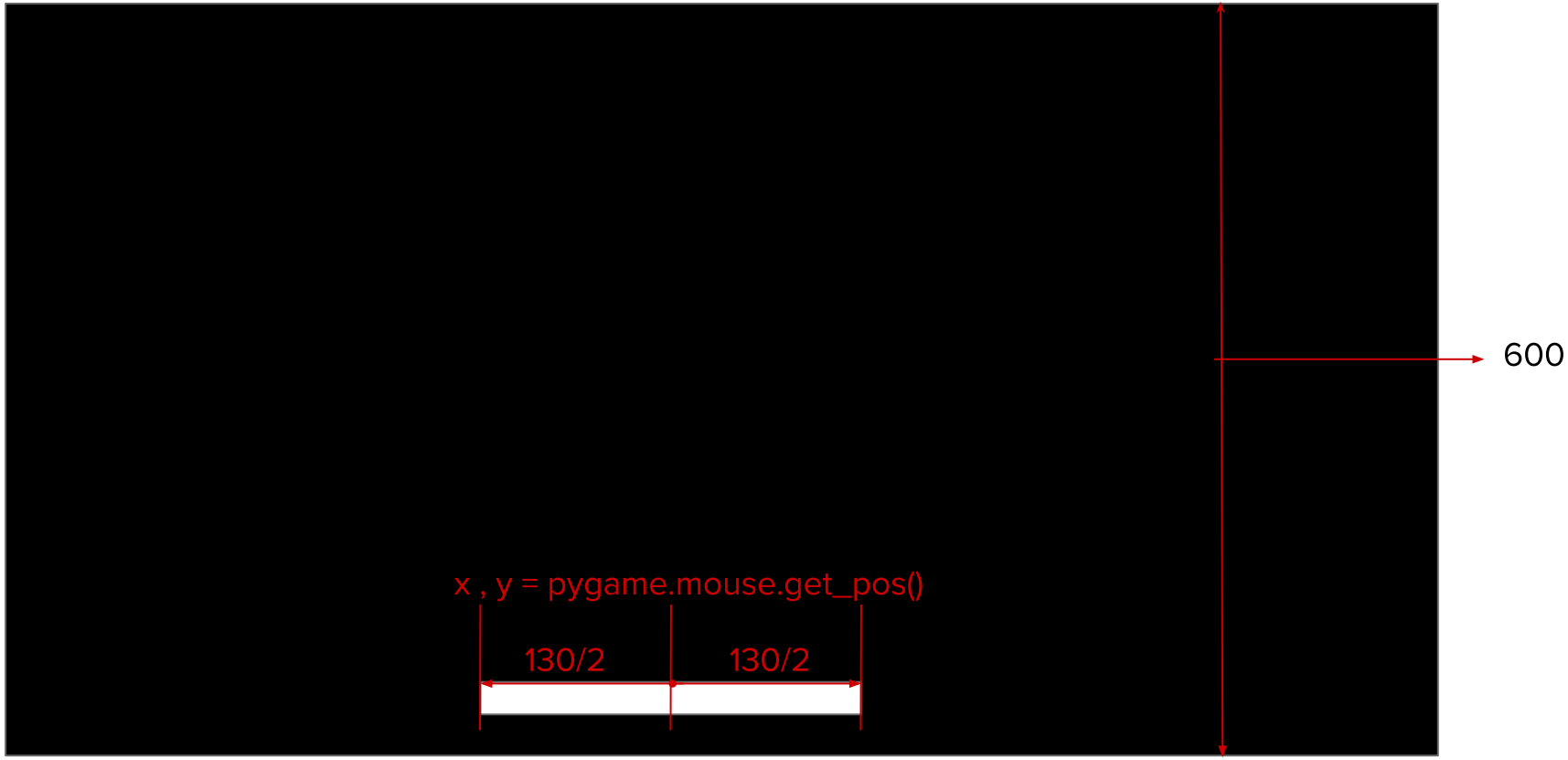
The ball will be bounce back even though they don't catch the rectangle.

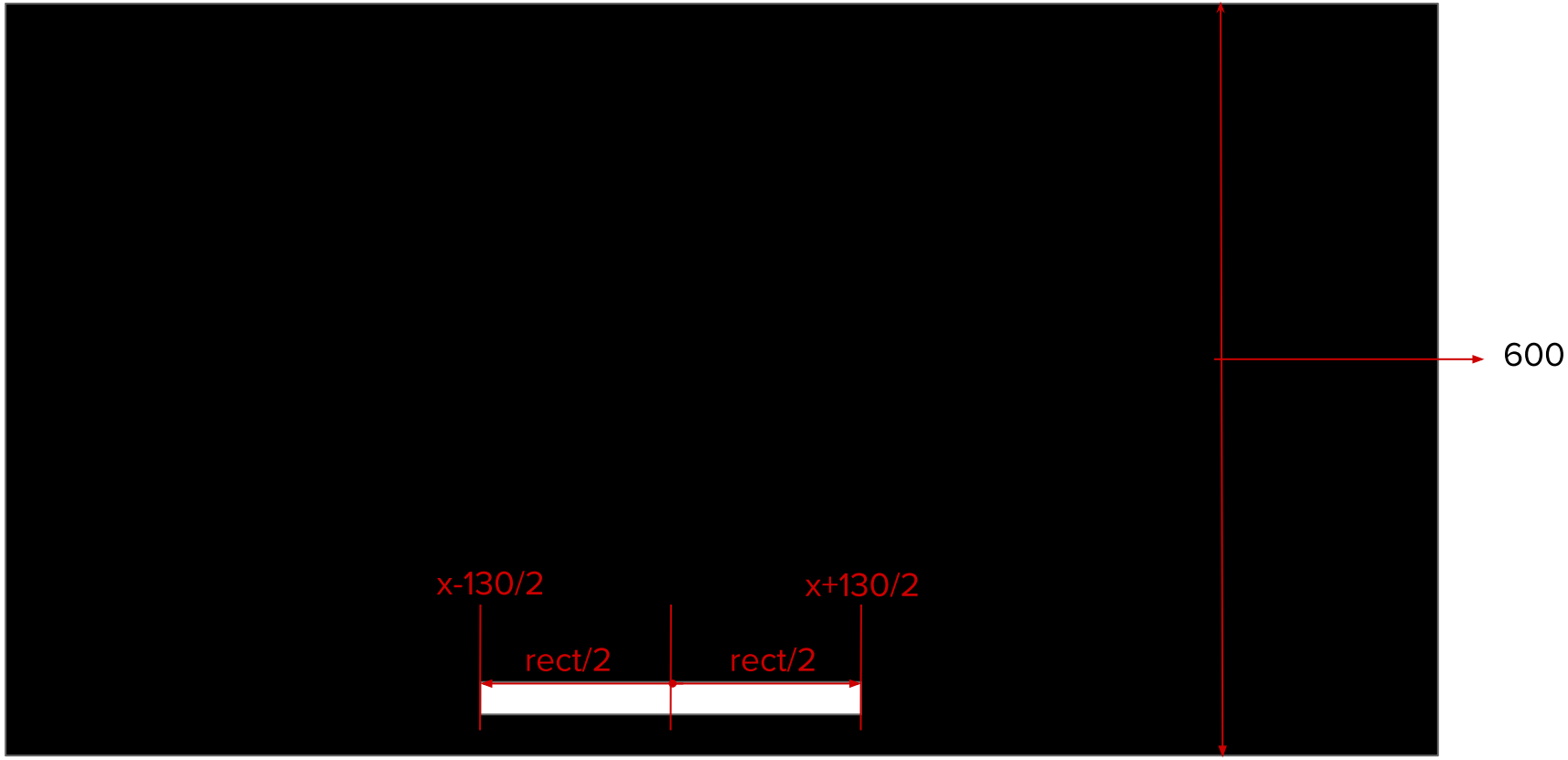
Always bounce back

at position_y = 580 - self.side

```
if self.rect.y > 580 - self.side:
    self.rect.y = 580 - self.side
    self.speed_y = -self.speed_y
```


600
x , y
rect/2
rect/2


600
x , y = pygame.mouse.get_pos()
130/2
130/2


600
x-130/2
x+130/2
rect/2
rect/2

More about rectangle

```
import pygame
import random

class Ball(pygame.sprite.Sprite):
...
    def update(self):
    ...
        if self.rect.y > 580 - self.side:
            x , y = pygame.mouse.get_pos()

            self.rect.y = 580 - self.side
            self.speed_y = -self.speed_y
...
pygame.quit()
```

To makes sure each ball bounces only when the ball is near the bar. In other word, x of the ball is not too far from the x of the mouse. Y is already taken care of because the initial if statement.

More about rectangle

```
import pygame
import random

class Ball(pygame.sprite.Sprite):
...
    def update(self):
    ...
        if self.rect.y > 580 - self.side:
            x , y = pygame.mouse.get_pos()
            if self.rect.x > x-130/2 and self.rect.x < x+130/2 :

                self.rect.y = 580 - self.side
                self.speed_y = -self.speed_y
...
pygame.quit()
```

Notice:
indentation
inside if
statement

observe...

It should be bouncing only when the bar is close enough now, but there is a little more we need to adjust.

The ball will bounce back when pass the rectangle

What happens if the ball passes 580 a little and the bar starts sliding towards it?
Think! Think! Think!

More about rectangle

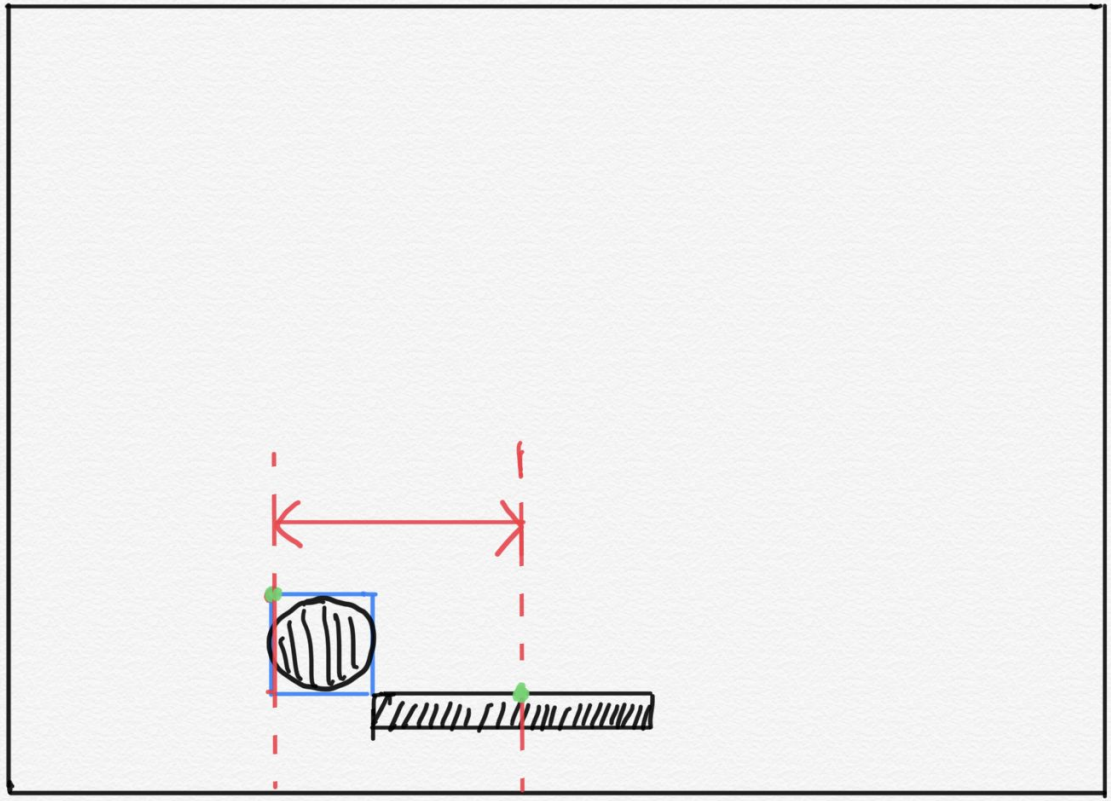
To fix this, we can simply add condition inside our if statement. So it can have a little gap where bouncing can still occurs. But if passed 580 for more than 10 pixels....

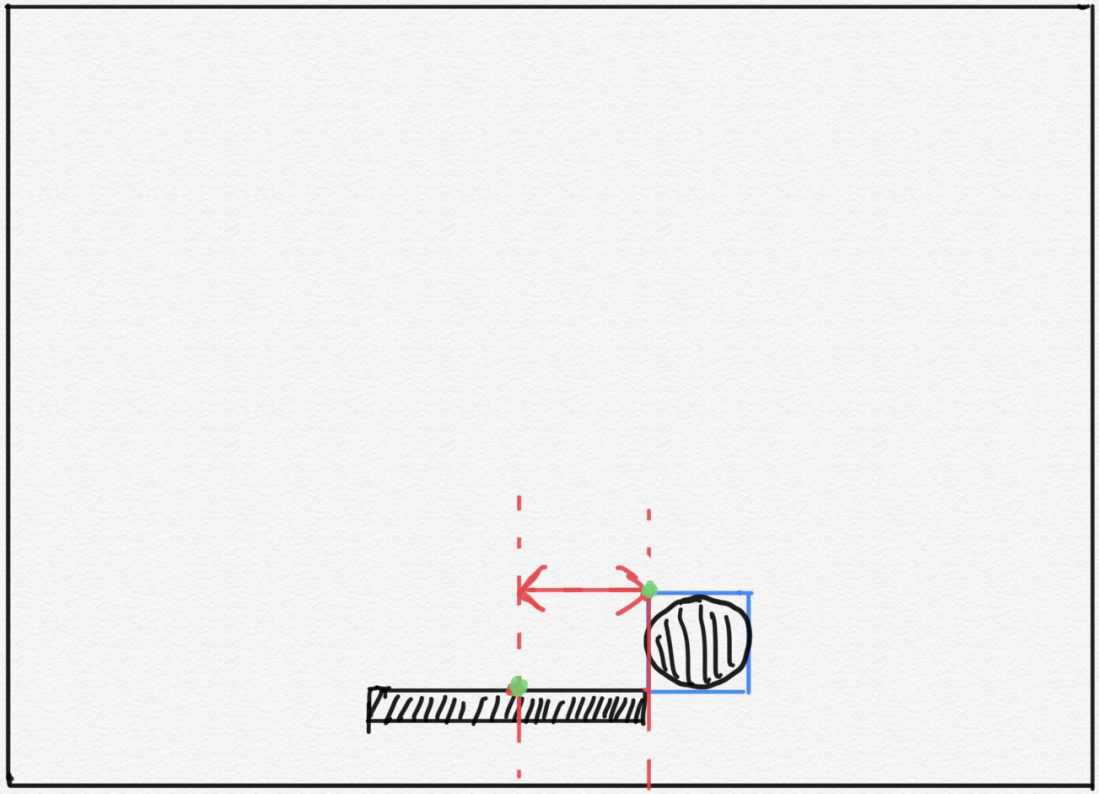
```
import pygame
import random

class Ball(pygame.sprite.Sprite):
...
    def update(self):
    ...
            if self.rect.y > 580 - self.side and self.rect.y <= 590 - self.side:
                x , y = pygame.mouse.get_pos()
                if self.rect.x > x-130/2 and self.rect.x < x+130/2 :
                    self.rect.y = 580 - self.side
                    self.speed_y = -self.speed_y
...
pygame.quit()
```

observe...

Do you notice we get different result from left and right side?

why?





Draw 2 lines

Why we need these 2 lines?

Draw 2 lines

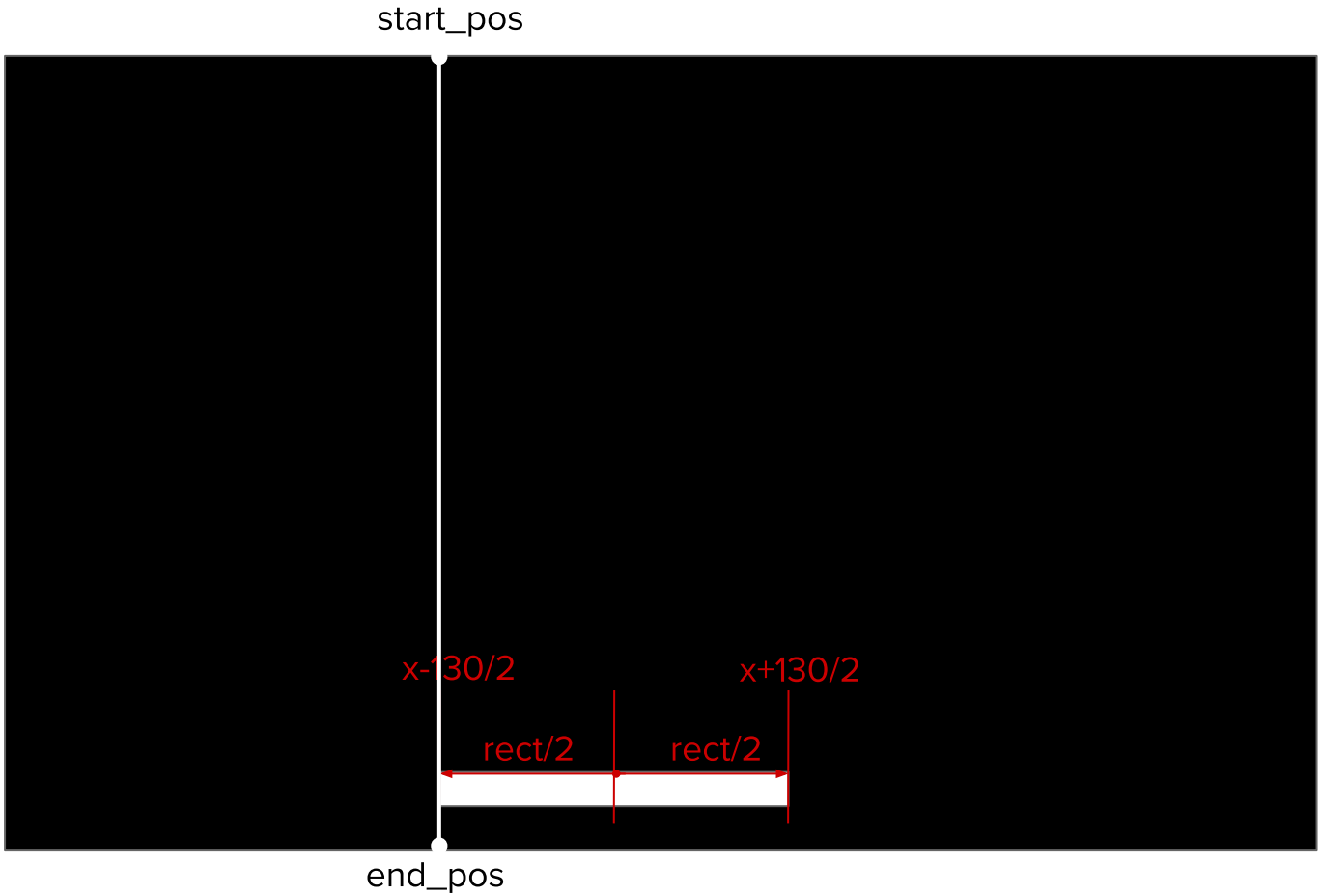
```
pygame.draw.line()
```

draw a straight line

```
line(surface, color, start_pos, end_pos, width) -> Rect
line(surface, color, start_pos, end_pos, width=1) -> Rect
```

Draw 2 lines

What is start_pos & end_pos?



Draw 2 lines

```
import pygame
...

done = False
while not done:
...
    x, y = pygame.mouse.get_pos()
    pygame.draw.rect(screen, (255,255,255), (x-130/2, 580, 130, 10))

    pygame.draw.line(screen, (255,255,255), (x-130/2, 0), (x-130/2, 800), 1)

    pygame.display.flip()
    allspriteslist.update()

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

Draw 2 lines

```
import pygame
...

done = False
while not done:
...
    x, y = pygame.mouse.get_pos()
    pygame.draw.rect(screen, (255,255,255), (x-130/2, 580, 130, 10))

    pygame.draw.line(screen, (255,255,255), (x-130/2, 0), (x-130/2, 800), 1)
    pygame.draw.line(screen, (255,255,255), (x+130/2, 0), (x+130/2, 800), 1)

    pygame.display.flip()
    allspriteslist.update()

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

More about bounce-back

```
import pygame
import random

class Ball(pygame.sprite.Sprite):
...
    def update(self):
    ...
                if self.rect.y > 580 - self.side and self.rect.y <= 590 - self.side:
                        x , y = pygame.mouse.get_pos()
                        if self.rect.x > x-130/2 - self.side and self.rect.x < x+130/2 :
                                self.rect.y = 580 - self.side
                                self.speed_y = -self.speed_y
...
pygame.quit()
```

The solution is rather simple. We just need to compensate the left side.

Creating new ball depends on time

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

    for event in pygame.event.get():

        if event.type == pygame.MOUSEBUTTONDOWN:
            createBall()
        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 and showIntro is True:
                showIntro = False
                start_time = pygame.time.get_ticks()
...
pygame.quit()
```

Let's create ball based on time. So let's disable the way we created a ball by clicking the mouse.

Creating new ball depends on time

-- lasttime_balladd variable

```
import pygame
...
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)

lasttime_balladd = pygame.time.get_ticks()

start_time = pygame.time.get_ticks()

time_left = 1000000
clicked_count = 0

showIntro = True
...
pygame.quit()
```

We need a variable to keep track of the timestamp when a ball was last created.

You might notice that is the same as start_time initially

Creating new ball depends on time

-- if statement

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

    # game playing
    elif remaining_time > 0:
        if pygame.time.get_ticks()-lasttime_balladd >5000:
            print("ball")
        allspriteslist.draw(screen)
        lasttime_balladd = pygame.time.get_ticks()

        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()
```

Create a new ball every n second

Creating new ball depends on time

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

    # game playing
    elif remaining_time > 0:
        if pygame.time.get_ticks()-lasttime_balladd >5000:
            print("ball")
            createBall()
            lasttime_balladd = pygame.time.get_ticks()
        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()
```

After we see "ball" being printed every 5 seconds, we can start calling the createBall() function again.

Creating new ball depends on time

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

    # game playing
    elif remaining_time > 0:
        if pygame.time.get_ticks()-lasttime_balladd >5000:
            createBall()
            lasttime_balladd = pygame.time.get_ticks()
        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()
```

Also we need to update the last time when a ball is added. We need to use the latest time.

challenge

Create the position of new ball randomly (Hint: modify the createBall() function)

Position of new ball

```
import pygame
...
def createBall():
    sc = random.randint(0, 255)
    x_pos, y_pos = pygame.mouse.get_pos()

    x_speed = random.randint(-3, 3)
    y_speed = random.randint(1, 3)

    x_pos = random.randint(0,800)
    y_pos = 150

    size = random.randint(5, 30)
    s = Ball(x_pos, y_pos, size, x_speed, y_speed, (sc, sc, sc))

    allspriteslist.add(s)

pygame.init() # 0 s
...
pygame.quit()
```

y_pos should remain the same so all the balls come down from the same height.

Where are the sprites?

Where are they after they left the screen? Are they really gone?

Remove the sprite when it's missed

Remove the sprite when it's missed

```
import pygame
import random

class Ball(pygame.sprite.Sprite):
...
    def update(self):
    ...
        if self.rect.y < 0:
            self.rect.y = 0
            self.speed_y = -self.speed_y
    ...

        if self.rect.y > 600:
            self.remove(allspriteslist)


allspriteslist = pygame.sprite.Group()
...
pygame.quit()
```

We need to add this self removing function inside the update. So it will remove itself from the group when if left the screen.

Remove the sprite when it's missed

```
import pygame
import random

class Ball(pygame.sprite.Sprite):
...
    def update(self):
    ...
        if self.rect.y < 0:
            self.rect.y = 0
            self.speed_y = -self.speed_y

        if self.rect.y > 600:
            self.remove(allspriteslist)


allspriteslist = pygame.sprite.Group()
...
pygame.quit()
```

It may cause error. If it does, move this line above the class.

```
import pygame
import random
import pygame.gfxdraw

allspriteslist = pygame.sprite.Group()

class Ball(pygame.sprite.Sprite):
```

Ball class might complain because allspriteslist might not yet have existed.

Remove the sprite when it's missed

```
import pygame
...
done = False
while not done:
...
    # game playing
    elif remaining_time > 0:
        if pygame.time.get_ticks()-lasttime_balladd >10000:
            createBall()
            lasttime_balladd = pygame.time.get_ticks()
        allspriteslist.draw(screen)
        print(allspriteslist.sprites())

        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()
```

Add this line during gameplay. To see the sprites inside our group.

Change time and observe whether it's removed

Remove the sprite when it's missed

```
import pygame
...
done = False
while not done:
...
    # game playing
    elif remaining_time > 0:
        if pygame.time.get_ticks()-lasttime_balladd >10000:
            createBall()
            lasttime_balladd = pygame.time.get_ticks()
        allspriteslist.draw(screen)
        print(allspriteslist.sprites())

        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()
```

Once we confirm it is good, we can remove it.