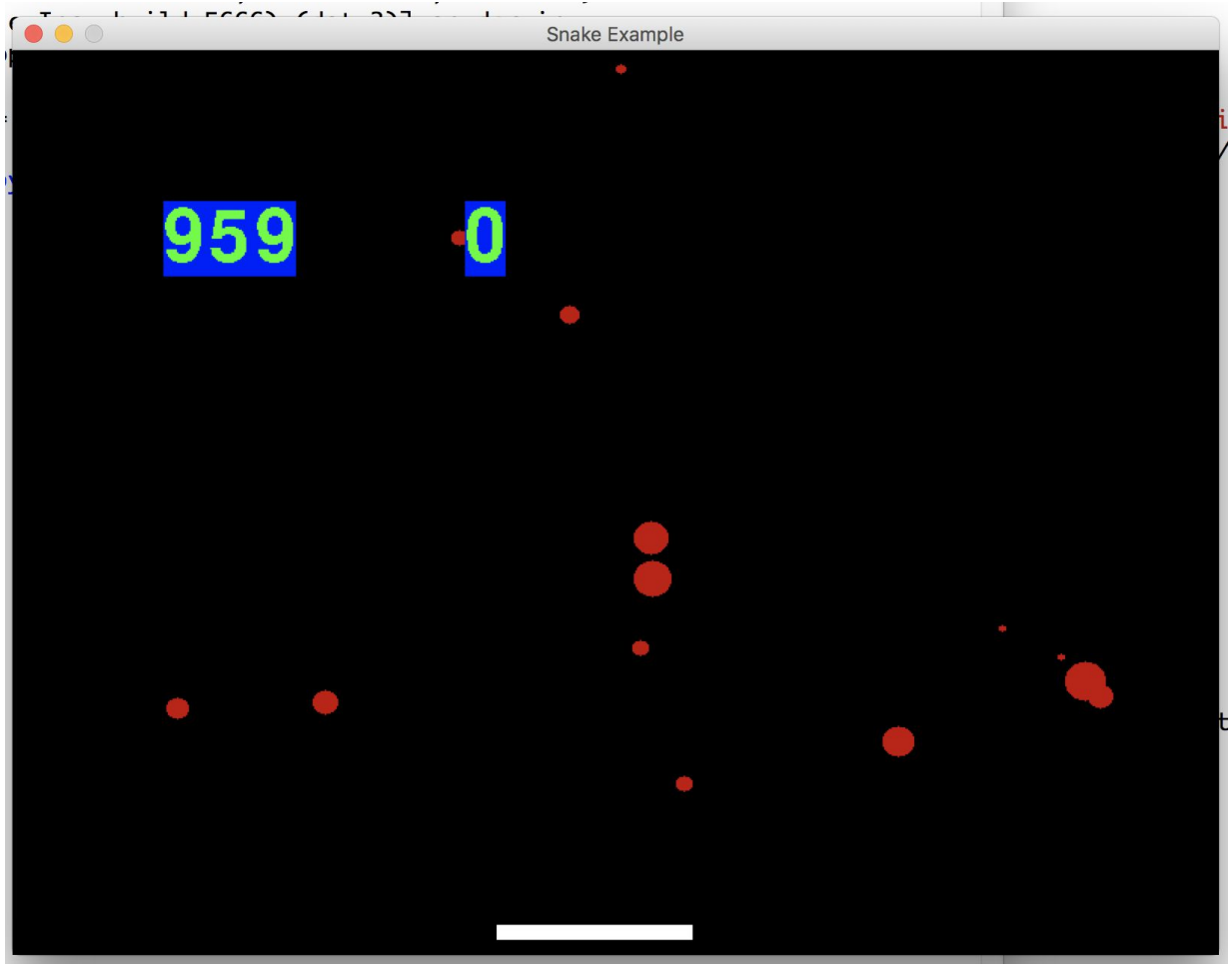
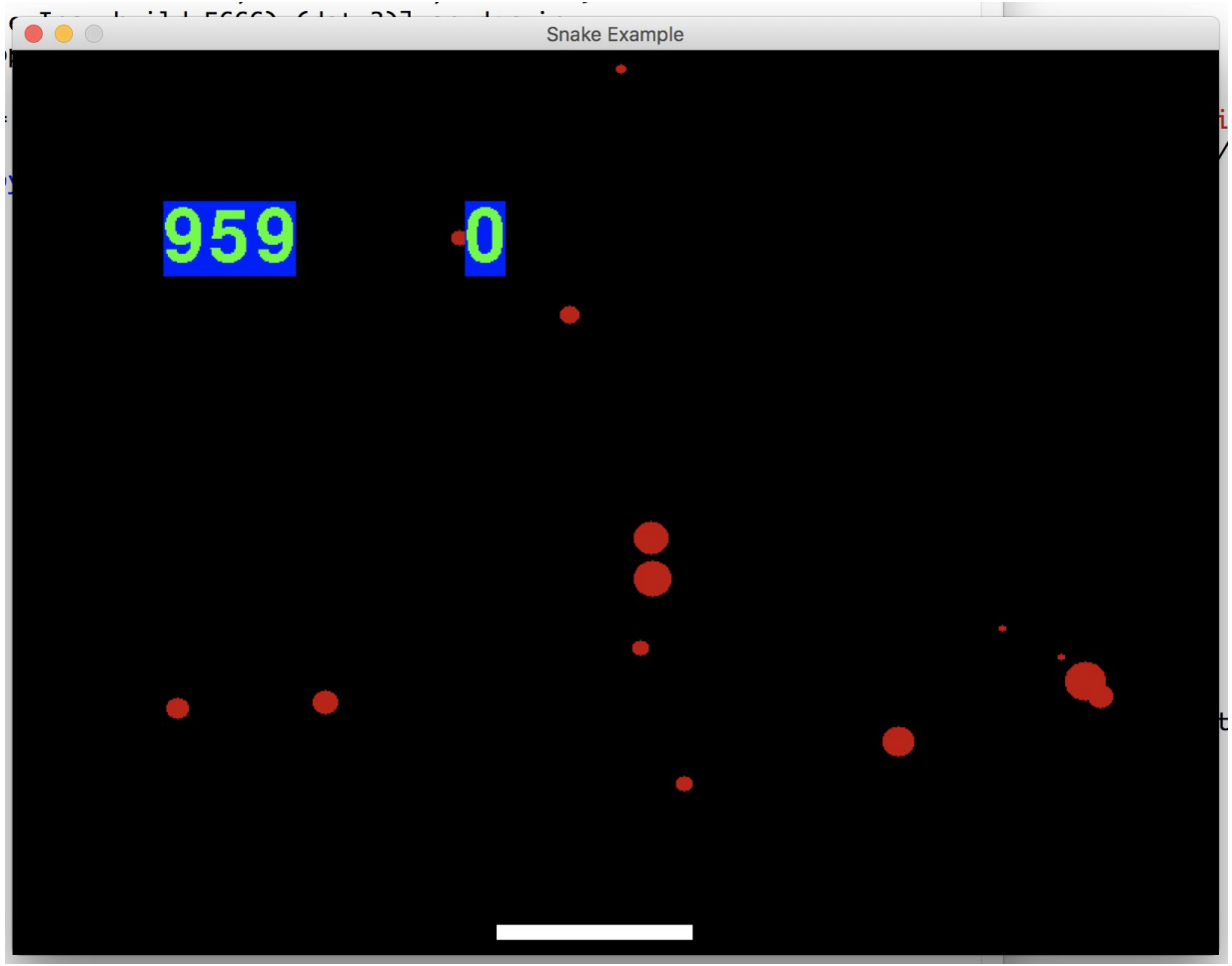
Game 2

Part 2

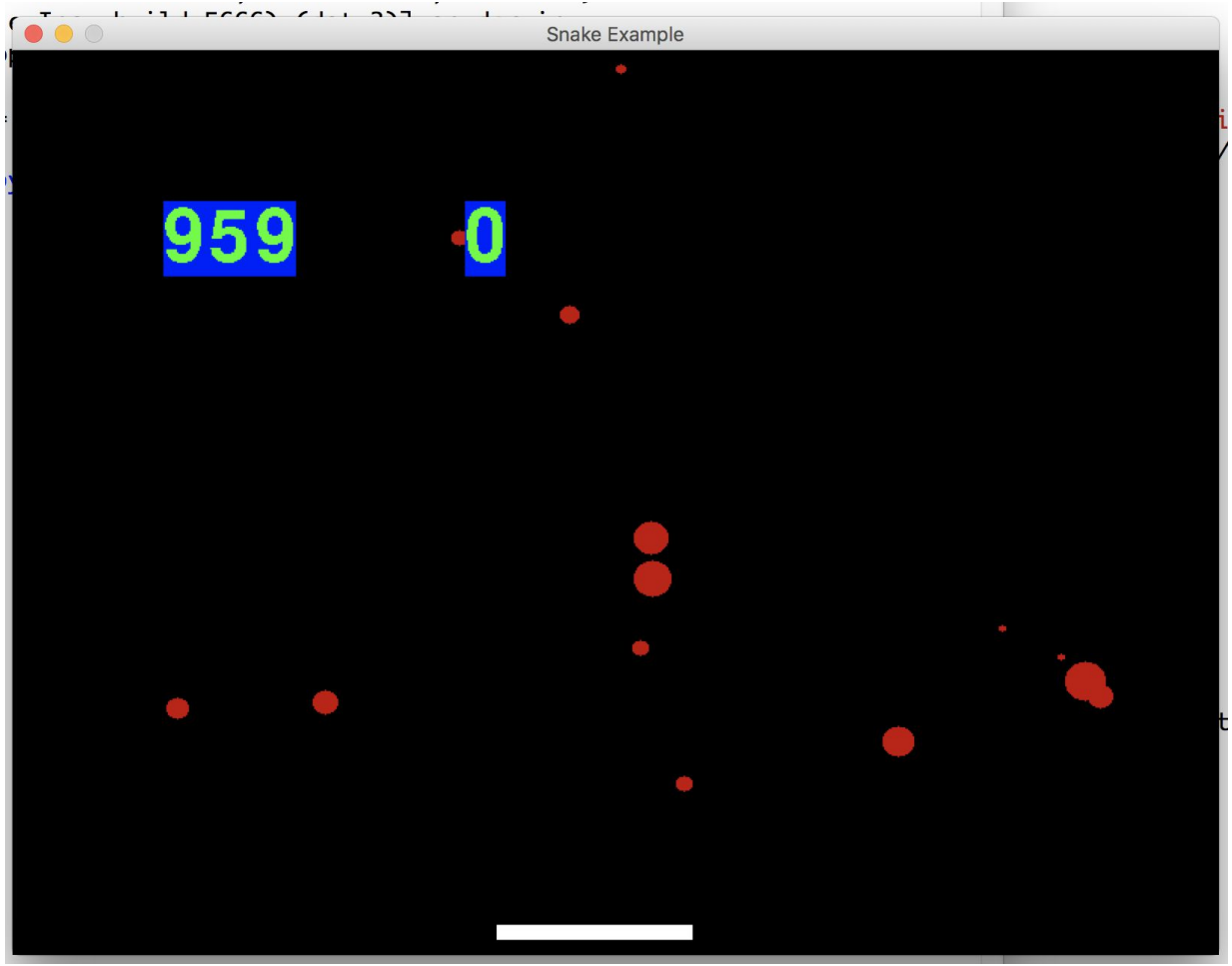
Job for today...



A white moving rectangle



The ball will Bounce back when touch the rectangle



replace class name ---- 2 places

```
import pygame
...

class Ball(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)
...
pygame.quit()
```

This time we open a new file but copy all the code from last week.

The first thing we do is to rename our class. We want to create a ball now. Not a Square anymore.

replace class name ---- 2 places

```
import pygame
...

allspriteslist = pygame.sprite.Group()

for i in range(60):
...

    x_speed = x * size/6.0
    y_speed = y * size/6.0


    s = Ball(x_pos, y_pos, size, x_speed, y_speed, (sc, sc, sc))

    allspriteslist.add(s)

pygame.init() # 0 s
```

This is the second place. Where we created the Squares is where we should create a Ball now.

Time to check your works,
Now everyone share your whole screen…

Bounce back ---- x

```
import pygame
...

class Ball(pygame.sprite.Sprite):
    ...
    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 = -self.side
            self.rect.x = 800 - self.side
            self.speed_x = -self.speed_x

        if self.rect.x < -self.side:
            self.speed_x = -self.speed_x

...
pygame.quit()
```

To make it bounces, we need to change the current behavior.

The detection needs to be changed too.
Before it detects when the Square is off the screen.

Now we want to check when it hits the right edge.

Observe...

Bounce back ---- x

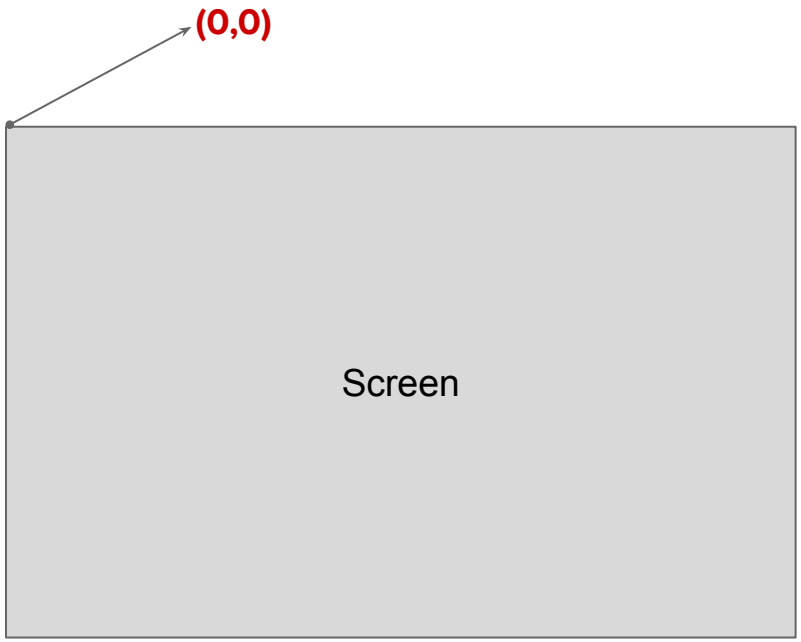
```
import pygame
...

class Ball(pygame.sprite.Sprite):
    ...
    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 = -self.side
            self.rect.x = 800 - self.side
            self.speed_x = -self.speed_x

        if self.rect.x < 0:
            self.speed_x = -self.speed_x

...
pygame.quit()
```



Bounce back ---- x

```
import pygame
...

class Ball(pygame.sprite.Sprite):
    ...
    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 = -self.side
            self.rect.x = 800 - self.side
            self.speed_x = -self.speed_x

        if self.rect.x < 0:
            self.rect.x = 0
            self.speed_x = -self.speed_x
...
pygame.quit()
```

For the left edge, the result is similar, but the detection is different. And remember it is always top left corner of the invisible rectangle wrapping the Ball we are using for detection.

Observe… and let’s work on the top and bottom

Bounce back ---- y

```
import pygame
...

class Ball(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=speed_y

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

...
pygame.quit()
```

To check the top and bottom.

Let's make the speed_x 0. And all the ball will be moving up either or down.

Bounce back ---- y

```
import pygame
...

class Ball(pygame.sprite.Sprite):
    ...
    def update(self):
    ...


        if self.rect.y > 600:
            self.speed_y = -self.speed_y

        if self.rect.y < -self.side:
            self.rect.y = 600

...
pygame.quit()
```

Similar to x, we just need to flip the speed from positive to negative.

Bounce back ---- y

```
import pygame
...

class Ball(pygame.sprite.Sprite):
    ...
    def update(self):
    ...


        if self.rect.y > 600-self.side:
            self.speed_y = -self.speed_y


        if self.rect.y < -self.side:
            self.rect.y = 600

...
pygame.quit()
```

For proper detection, we will need to adjust the if statement.

Bounce back ---- y

```
import pygame
...

class Ball(pygame.sprite.Sprite):
    ...
    def update(self):
    ...


        if self.rect.y > 600-self.side:
            self.rect.y = 600 - self.side
            self.speed_y = -self.speed_y


        if self.rect.y < -self.side:
            self.rect.y = 600

...
pygame.quit()
```

To make sure there is no chance, not even a single frame that the Ball can leave off the screen by 1 pixel.

We simply force it to be within the screen.

Bounce back ---- y

```
import pygame
...

class Ball(pygame.sprite.Sprite):
    ...
    def update(self):
    ...


        if self.rect.y > 600-self.side:
            self.rect.y = 600 - self.side
            self.speed_y = -self.speed_y


        if self.rect.y < -self.side:
            self.speed_y = -self.speed_y

...
pygame.quit()
```

For bottom side

Bounce back ---- y

```
import pygame
...

class Ball(pygame.sprite.Sprite):
    ...
    def update(self):
    ...


        if self.rect.y > 600-self.side:
            self.rect.y = 600 - self.side
            self.speed_y = -self.speed_y


        if self.rect.y < 0:
            self.speed_y = -self.speed_y

...
pygame.quit()
```

Does this need to be adjusted?

Bounce back ---- y

```
import pygame
...

class Ball(pygame.sprite.Sprite):
    ...
    def update(self):
    ...


        if self.rect.y > 600-self.side:
            self.rect.y = 600 - self.side
            self.speed_y = -self.speed_y


        if self.rect.y < 0:
            self.rect.y = 0
            self.speed_y = -self.speed_y

...
pygame.quit()
```

Again force it within the screen.

Bounce back ---- y

```
import pygame
...

class Ball(pygame.sprite.Sprite):
    ...
    def update(self):
    ...

        if self.rect.y > 600-self.side:
            self.rect.y = 600 - self.side
            self.speed_y = -self.speed_y


        if self.rect.y < = 0:
            self.rect.y = 0
            self.speed_y = -self.speed_y

...
pygame.quit()
```

Q: why “=” sign?

Better yet, we will add an equal sign in the if statement.

Think about it frame by frame.

Bounce back all ---- x & y

```
import pygame
...

class Ball(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=speed_y

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


...
pygame.quit()
```

Bounce back all ---- x & y

```
import pygame
...

for i in range(60):
    sc = random.randint(0, 255)
    x_pos = random.randint(0, 800)
    y_pos = random.randint(0, 600)
    size = random.randint(5, 30)

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

    x_speed = x * size/6.0
    y_speed = y * size/6.0

...
pygame.quit()
```

Sort your if statement

```
done = False
while not done:
    ...
    # intro screen
    if showIntro is True:
        intro_text = font.render("Welcome", False, (0, 255, 0))
        screen.blit(intro_text, (100, 100))
    # game over
    elif 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()
    # game playing
    else:
        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))
```

It is time to rearrange the order of your game flow. To make it more intuitive, we will put the game-over block at the end.

Sort your if statement

```
done = False
while not done:
    ...
    # intro screen
    if showIntro is True:
        intro_text = font.render("Welcome", False, (0, 255, 0))
        screen.blit(intro_text, (100, 100))
    # game playing
    elif remaining_time <= 0:
        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))
    # game over
    else:
        remaining_time = 0
        over_text = font.render("Game Over", False, (0, 255, 0), (0, 0, 255))
        screen.blit(over_text, (200,300))
        allspriteslist.empty()
```

Sort your if statement

```
done = False
while not done:
    ...
    # intro screen
    if showIntro is True:
        intro_text = font.render("Welcome", False, (0, 255, 0))
        screen.blit(intro_text, (100, 100))
    # game playing
    elif remaining_time > 0:
        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))
    # game over
    else:
        remaining_time = 0
        over_text = font.render("Game Over", False, (0, 255, 0), (0, 0, 255))
        screen.blit(over_text, (200,300))
        allspriteslist.empty()
```

Change the less than or equal sign <= to greater than >
Why not >= ?

Review your logic
Validate
Confirm

Create balls when you click ---- change for loop

```
def createBall():
    sc = random.randint(0, 255)
    x_pos = random.randint(0, 800)
    y_pos = random.randint(0, 600)
    size = random.randint(5, 30)

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

    x_speed = x * size/6.0
    y_speed = y * size/6.0

    s = Ball(x_pos, y_pos, size, x_speed, y_speed, (sc, sc, sc))

    allspriteslist.add(s)
```

Change the for loop into a function

Create balls when you click ---- use function

```
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()
```

Capture one more event.

MOUSEBUTTONDOWN

Create balls where you click ---- function

```
def createBall():
    sc = random.randint(0, 255)
    x_pos, y_pos = pygame.mouse.get_pos()
    size = random.randint(5, 30)

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

    x_speed = x * size/6.0
    y_speed = y * size/6.0

    s = Ball(x_pos, y_pos, size, x_speed, y_speed, (sc, sc, sc))

    allspriteslist.add(s)

pygame.init() # 0 s
```

Tuple

When a function returns a Tuple. We can receive with two variable.

`pygame.mouse.get_pos()`

get the mouse cursor position

`get_pos() -> (x, y)`

Returns the `x` and `y` position of the mouse cursor. The position is relative to the top-left corner of the display. The cursor position can be located outside of the display window, but is always constrained to the screen.

Search examples for pygame.mouse.get_pos | Comments 1

Now Change speed_x, speed_y randomly...

Change speed ---- x

```
def createBall():
    sc = random.randint(0, 255)
    size = random.randint(5, 30)
    x_pos, y_pos = pygame.mouse.get_pos()

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

    x_speed = x * size/6.0
    y_speed = y * size/6.0

    s = Ball(x_pos, y_pos, size, x_speed, y_speed, (sc, sc, sc))

    allspriteslist.add(s)
```

What this part working for?

Change speed ---- x

```
def createBall():
    sc = random.randint(0, 255)
    size = random.randint(5, 30)
    x_pos, y_pos = pygame.mouse.get_pos()

    x_speed = random.randint(-3, 3)

    x = 0
    while x == 0:
        x = random.randint( 0, 1 )
    y = 0
    while y == 0:
        y = random.randint( 0, 1 )
    x_speed = x * size/6.0
    y_speed = y * size/6.0

    s = Ball(x_pos, y_pos, size, x_speed, y_speed, (sc, sc, sc))

    allspriteslist.add(s)
```

Change speed ---- y

```
import pygame
...

def createBall():
    sc = random.randint(0, 255)
    size = random.randint(5, 30)
    x_pos, y_pos = pygame.mouse.get_pos()

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

    y = 0
    while y == 0:
        y = random.randint( 0, 1 )
    y_speed = y * size/6.0

    s = Ball(x_pos, y_pos, size, x_speed, y_speed, (sc, sc, sc))
    allspriteslist.add(s)
...
pygame.quit()
```

Change speed ---- y

```
import pygame
...

def createBall():
    sc = random.randint(0, 255)
    size = random.randint(5, 30)
    x_pos, y_pos = pygame.mouse.get_pos()

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


    s = Ball(x_pos, y_pos, size, x_speed, y_speed, (sc, sc, sc))

    allspriteslist.add(s)

...
pygame.quit()
```

Why always positive?

For the bar at the bottom.

On which Surface should we Draw a rectangle?

Draw a rectangle on screen Surface

```
pygame.draw.rect()
```

draw a rectangle

```
rect(surface, color, rect) -> Rect
rect(surface, color, rect, width=0, border_radius=0,
border_radius=-1, border_top_left_radius=-1,
border_top_right_radius=-1, border_bottom_left_radius=-1) -> Rect
```

Draws a rectangle on the given surface.

```
pygame.draw.rect(surface, color, rect)
```

pygame.draw.rect(**screen**, (255,255,255), (x-130/2, 580, 130, 10))

Draw a rectangle on screen Surface

```
pygame.draw.rect()
```

draw a rectangle

```
rect(surface, color, rect) -> Rect
rect(surface, color, rect, width=0, border_radius=0,
border_radius=-1, border_top_left_radius=-1,
border_top_right_radius=-1, border_bottom_left_radius=-1) -> Rect
```

Draws a rectangle on the given surface.

```
pygame.draw.rect(surface, color, rect)
```

pygame.draw.rect(screen, **(255,255,255)**, (x-130/2, 580, 130, 10))

Draw a rectangle on screen Surface

```
pygame.draw.rect()
```

draw a rectangle

```
rect(surface, color, rect) -> Rect
rect(surface, color, rect, width=0, border_radius=0,
border_radius=-1, border_top_left_radius=-1,
border_top_right_radius=-1, border_bottom_left_radius=-1) -> Rect
```

Draws a rectangle on the given surface.

```
pygame.draw.rect(surface, color, rect)
```

rect (***Rect***) -- rectangle to draw, position and dimensions

pygame.draw.rect(screen, (255,255,255), **(0,0, 130, 10)**)

Length and width

Draw a rectangle on screen Surface

```
pygame.draw.rect()
```

draw a rectangle

```
rect(surface, color, rect) -> Rect
rect(surface, color, rect, width=0, border_radius=0,
border_radius=-1, border_top_left_radius=-1,
border_top_right_radius=-1, border_bottom_left_radius=-1) -> Rect
```

Draws a rectangle on the given surface.

```
pygame.draw.rect(surface, color, rect)
```

rect (***Rect***) -- rectangle to draw, position and dimensions

pygame.draw.rect(screen, (255,255,255), **(0,0, 130, 10)**)

position

Draw a rectangle on screen Surface

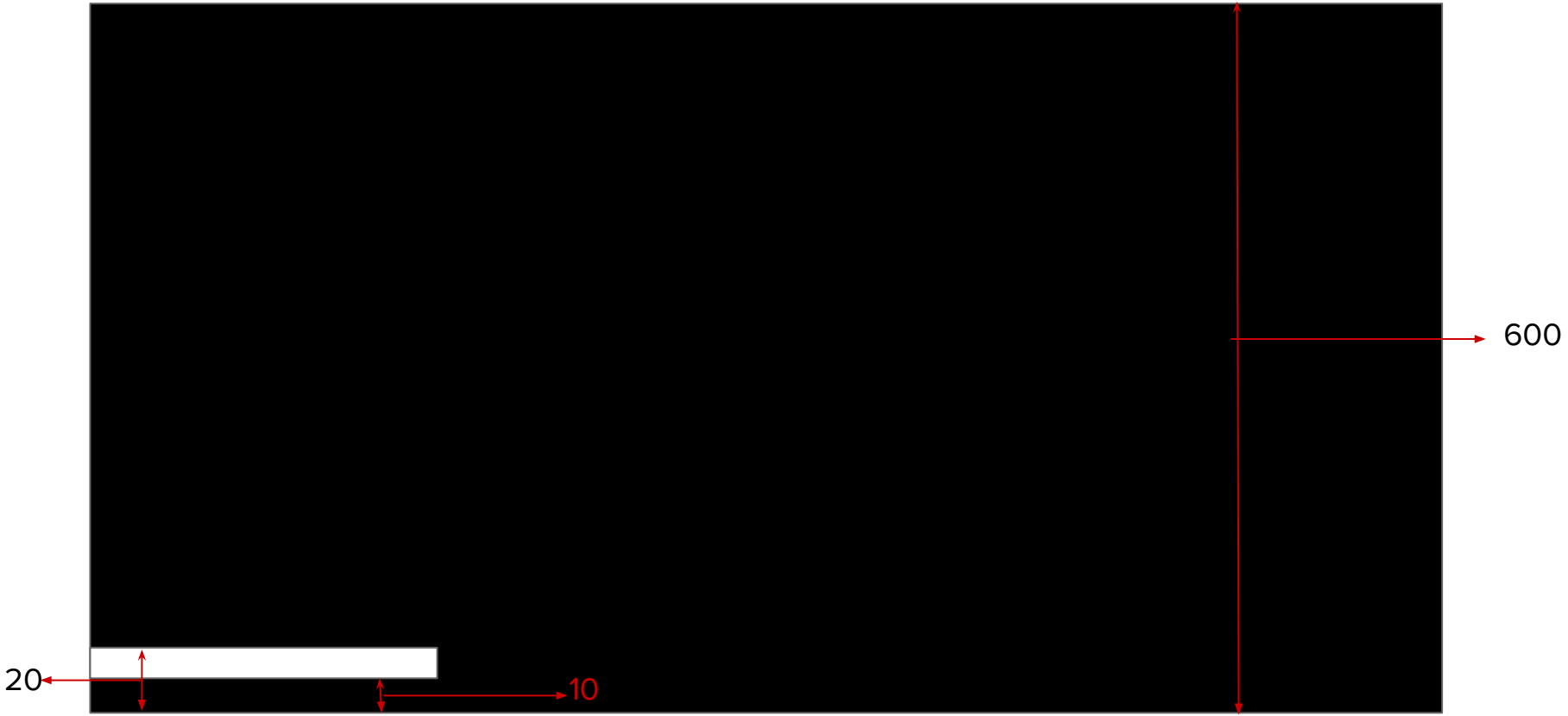
```
import pygame
...

done = False
while not done:
...

    pygame.draw.rect(screen, (255,255,255), (0,0, 130, 10))

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

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


600
20
10

Draw a rectangle on screen Surface

```
import pygame
...

done = False
while not done:
...

    pygame.draw.rect(screen, (255,255,255), (0, 580, 130, 10))


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

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

move rectangle as your mouse

Draw a rectangle on screen Surface

```
import pygame
...

done = False
while not done:
...

    x, y = pygame.mouse.get_pos()
    pygame.draw.rect(screen, (255,255,255), (x, 580, 130, 10))

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

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

observe...

Draw a rectangle on screen Surface

```
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.display.flip()
    allspriteslist.update()

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

Why 130/2 ?