プログラミング教室のテクノロ

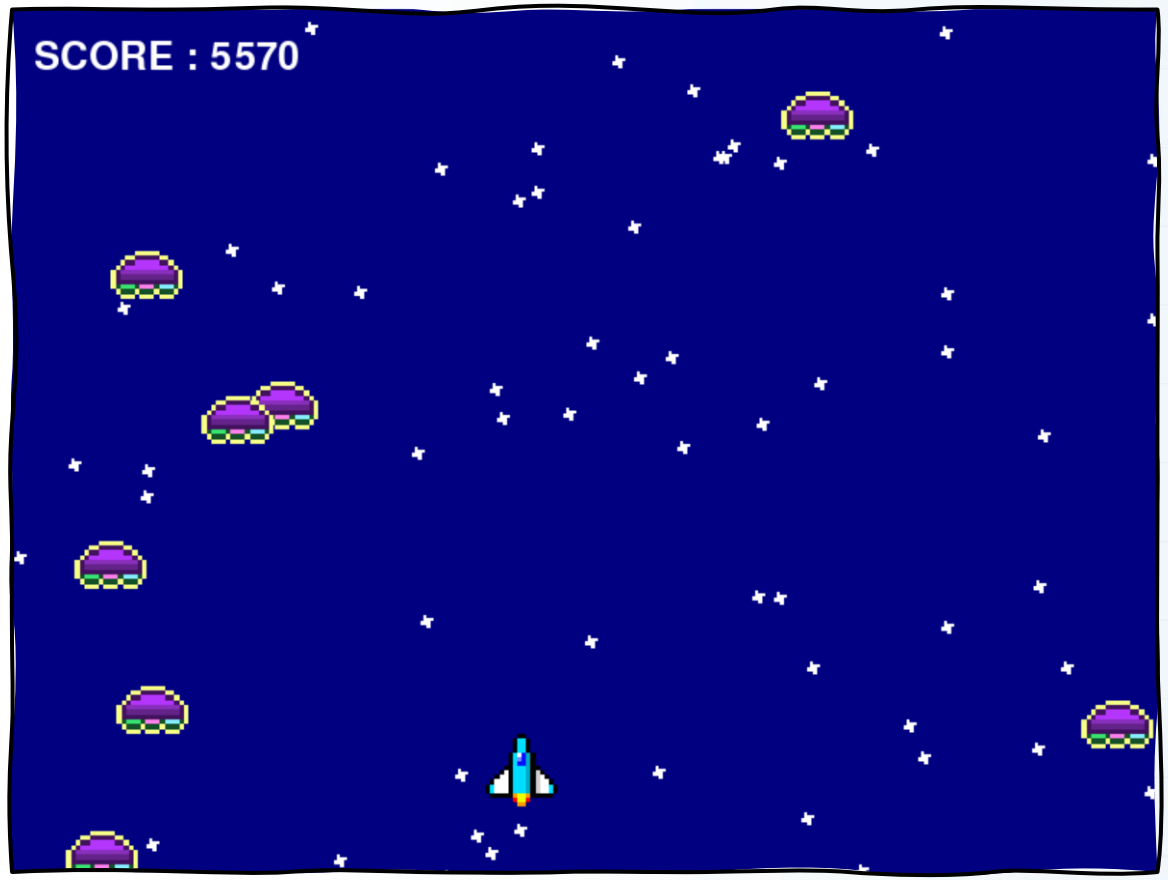
Pythonの道

pygameの使い方⑥

もくじ

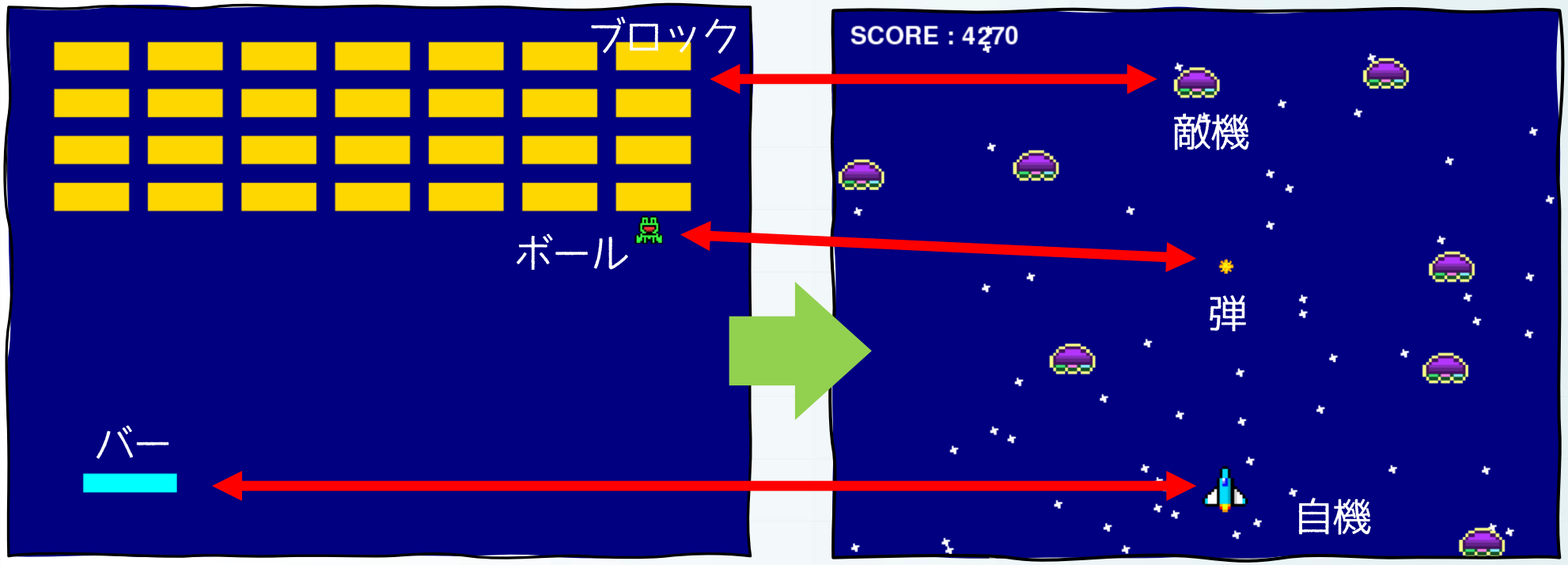
・pygameの使い方⑥

pygameを使って本格的なゲーム開発を学ぶ



シューティングゲームを作る

ブロック崩しゲームを改造してシューティングゲームを作る。



「シューティングゲーム」と「ブロック崩しゲーム」は似ている。「バー」と「ボール」と「ブロック」を「自機」と「弾」と「敵機」に置き換えることで「ブロック崩しゲーム」の仕組みを利用してシューティングゲームを作ることができる。

シューティングゲームを作る

```
#ゲームの準備をする
import pygame as pg, sys
import random
pg.init()
screen = pg.display.set_mode((800, 600))
#自機データ
myimg = pg.image.load("images/myship.png")
myimg = pg.transform.scale(myimg, (50, 50))
myrect = pg.Rect(400, 500, 50, 50)
#弾データ
bulletimg = pg.image.load("images/bullet.png")
bulletimg = pg.transform.scale(bulletimg, (16, 16))
bulletrect = pg.Rect(400, -100, 16, 16)
#UFOデータ
ufoimg = pg.image.load("images/UFO.png")
```

・・画像サイズの変更
transform.scale（画像変数,(幅,高さ）

次のページに続く

シューティングゲームを作る

```
ufoimg = pg.transform.scale(ufoimg, (50, 50))
ufos = []
for i in range(10):
    ux = random.randint(0,800)
    uy = -100 * i
    ufos.append(pg.Rect(ux, uy, 50, 50))
#星データ
starimg = pg.image.load("images/star.png")
starimg = pg.transform.scale(starimg, (12, 12))
stars = []
for i in range(60):
    star = pg.Rect(random.randint(0,800), 10 * i, 30, 30)
    star.w = random.randint(5,8)  ・・・星のy座標にいれる変数
    stars.append(star)
#ボタンデータ
```

・・・解説1

次のページに続く

シューティングゲームを作る

```
replay_img = pg.image.load("images/replaybtn.png")
#メインループで使う変数
pushFlag = False
page = 1
score = 0
#btnを押したら、newpageにジャンプする
def button_to_jump(btn, newpage):
    global page, pushFlag
    #ユーザーからの入力を調べる
    mdown = pg.mouse.get_pressed()
    (mx, my) = pg.mouse.get_pos()
    if mdown[0]:
        pg.mixer.Sound("sounds/pi.wav").play()
        if btn.collidepoint(mx, my) and pushFlag == False:
            page = newpage
```

次のページに続く

シューティングゲームを作る

```
                pushFlag = True
    else:
        pushFlag = False
#ゲームステージ
def gamestage():
    #画面を初期化する
    global score
    global page
    screen.fill(pg.Color("NAVY"))
    #ユーザーからの入力を調べる
    (mx, my) = pg.mouse.get_pos()
    mdown = pg.mouse.get_pressed()
    #絵を描いたり、判定したりする
    #星の処理
    for star in stars:
```

次のページに続く

シューティングゲームを作る

```
        star.y += star.w
        screen.blit(starimg, star)
        if star.y > 600:
            star.x = random.randint(0,800)
            star.y = 0
    #自機の処理
    myrect.x = mx - 25  ・・・解説2
    screen.blit(myimg, myrect)
    #弾の処理
    if mdown[0] and bulletrect.y < 0:
        bulletrect.x = myrect.x + 25 - 8
        bulletrect.y = myrect.y
        pg.mixer.Sound("sounds/beam.wav").play()
    if bulletrect.y >= 0:
        bulletrect.y += -15
```

・・・解説3

・・・弾のY座標が０より大きい間、Y座標を-15ずつ、変化させる

次のページに続く

シューティングゲームを作る

```
        screen.blit(bulletimg, bulletrect)
    #UFOの処理
    for ufo in ufos:
        ufo.y += 10
        screen.blit(ufoimg, ufo)
        if ufo.y > 600:
            ufo.x = random.randint(0,800)
            ufo.y = -100
        #自機とUFOの衝突処理
        if ufo.colliderect(myrect):
            page = 2
            pg.mixer.Sound("sounds/down.wav").play()
        #弾とUFOの衝突処理
        if ufo.colliderect(bulletrect):
            score = score + 1000
```

・・・解説1

次のページに続く

シューティングゲームを作る

```
                ufo.y = -100
                ufo.x = random.randint(0,800)
                bulletrect.y = -100
                pg.mixer.Sound("sounds/piko.wav").play()
    #スコアの処理
    score = score + 10
    font = pg.font.Font(None, 40)
    text = font.render("SCORE : "+str(score), True, pg.Color("WHITE"))
    screen.blit(text, (20, 20))
#データのリセット
def gamereset():
    global score
    score = 0
    myrect.x = 400
    myrect.y = 500
```

次のページに続く

シューティングゲームを作る

```
    bulletrect.y = -100
    for i in range(10):
        ufos[i] = pg.Rect(random.randint(0,800), -100 * i, 50, 50)
#ゲームオーバー
def gameover():
    screen.fill(pg.Color("NAVY"))
    font = pg.font.Font(None, 150)
    text = font.render("GAMEOVER", True, pg.Color("RED"))
    screen.blit(text, (100, 200))
    btn1 = screen.blit(replay_img,(320, 480))
    font = pg.font.Font(None, 40)
    text = font.render("SCORE : "+str(score), True, pg.Color("WHITE"))
    screen.blit(text, (20, 20))
    #絵を描いたり、判定したりする
    button_to_jump(btn1, 1)
```

・・・文字列の画像を作る関数

次のページに続く

シューティングゲームを作る

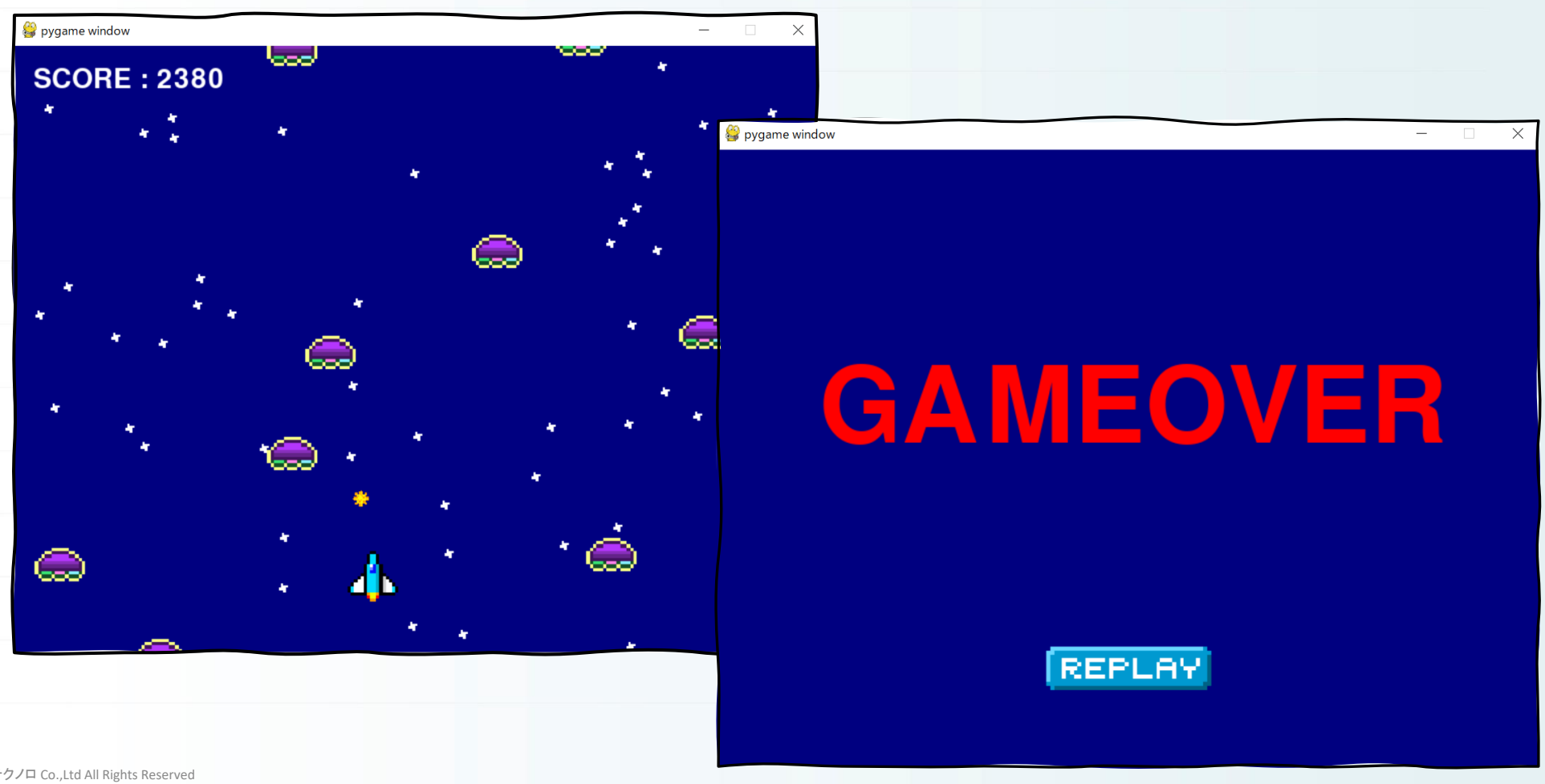
```
    #ボタンを押してリプレイしたら、ゲームをリセット
    if page == 1:
        gamereset()
#この下をずっとループする
while True:
    if page == 1:
        gamestage()
    elif page == 2:
        gameover()
    #画面を表示する
    pg.display.update()
    pg.time.Clock().tick(60)
    #閉じるボタンが押されたら、終了する
    for event in pg.event.get():
        if event.type == pg.QUIT:
```

次のページに続く

シューティングゲームを作る

```
pg.quit()
sys.exit()
```

「Run Module」またはF5キーを押してプログラムを実行する。



解説1:UFOの位置について

UFOを画面に複数表示するために空っぽのリストを作成し、
for文を使ってリストの中にUFOの座標を代入する。

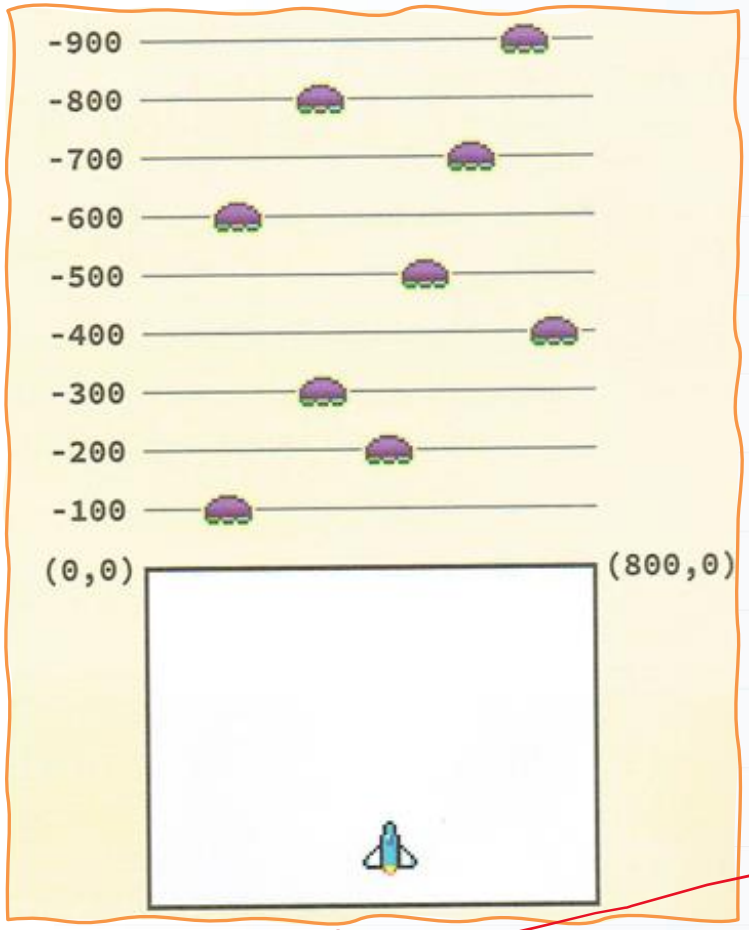
```
import random
import pygame as pg

ufos = []  ・・・空のリストを作る
for i in range(10):
    ux = random.randint(0,800)・・・UFOのｘ座標　0~800までの乱数が入る
    uy = -100 * i                ・・・UFOのy座標　-100 ~ -900の値が入る
    ufos.append(pg.Rect(ux, uy, 50, 50))   ・・・リストに要素を追加する
print(ufos)
```

【例】
リスト=[“A”, “B”, “C”]
リスト.append(“D”)
⇒ ['A', 'B', 'C', 'D']

```
[<rect(618, 0, 50, 50)>, <rect(242, -100, 50, 50)>, <
rect(219, -200, 50, 50)>, <rect(767, -300, 50, 50)>,
<rect(228, -400, 50, 50)>, <rect(165, -500, 50, 50)>,
<rect(563, -600, 50, 50)>, <rect(188, -700, 50, 50)>,
<rect(120, -800, 50, 50)>, <rect(54, -900, 50, 50)>]
```

解説1:UFOの位置について

uyに０～ -900の値が代入されることでUFOの座標が決定する



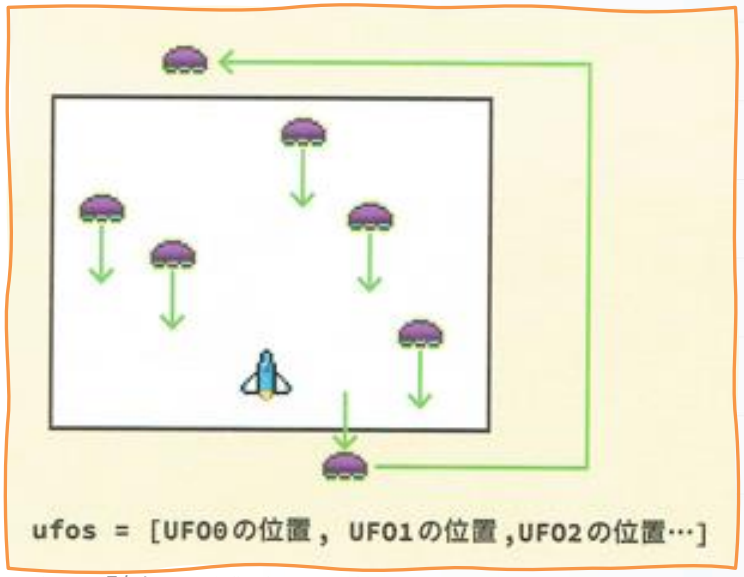
uy

```
[<rect(618,   0,  50,  50)>, <rect(242,  -100,  50,  50)>, <
rect(219,  -200,  50,  50)>, <rect(767,  -300,  50,  50)>,
<rect(228,  -400,  50,  50)>, <rect(165,  -500,  50,  50)>,
<rect(563,  -600,  50,  50)>, <rect(188,  -700,  50,  50)>,
<rect(120,  -800,  50,  50)>, <rect(54,  -900,  50,  50)>]
```

解説1:UFOの位置について

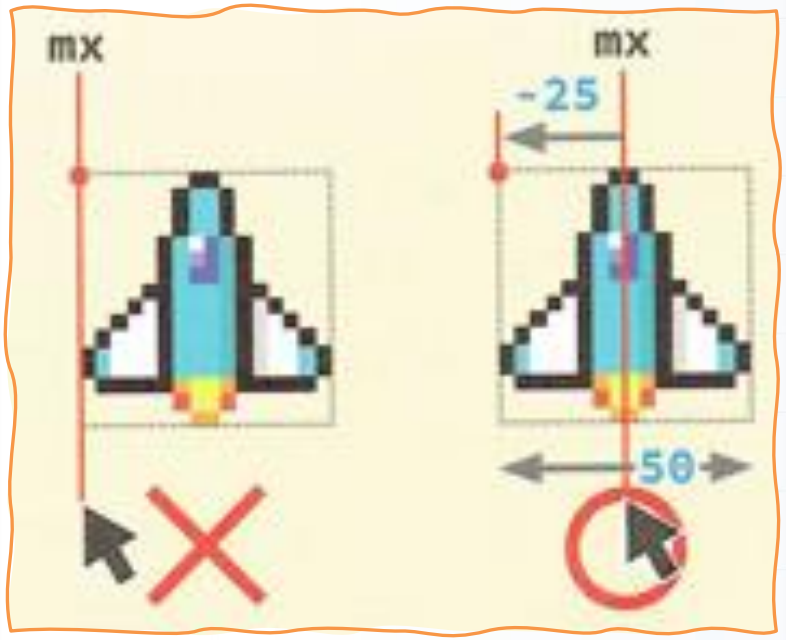
```
for ufo in ufos:
    ufo.y += 10
    screen.blit(ufoimg, ufo)
    if ufo.y > 600:
        ufo.x = random.randint(0,800)
        ufo.y = -100
```

リスト（ufos）の〇番目のｙ座標を10ずつ変えることでＵＦＯの位置が下に降りてくる。



解説2:自機の位置について

自機をマウスのｘ座標から25(自機の幅の半分)を引いた位置にする。マウス操作の際、自機の真ん中でコントロールすることができる。

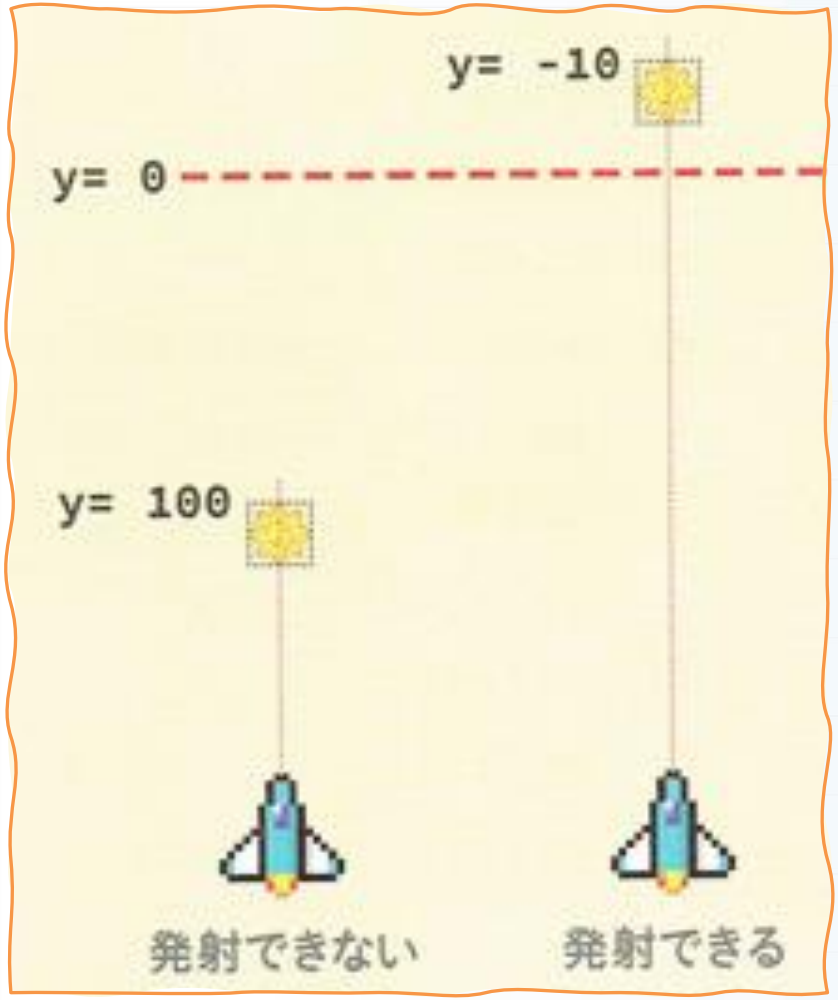


マウス座標の端に自機がくる

マウス座標の真ん中に自機がくる

解説3:弾の処理について

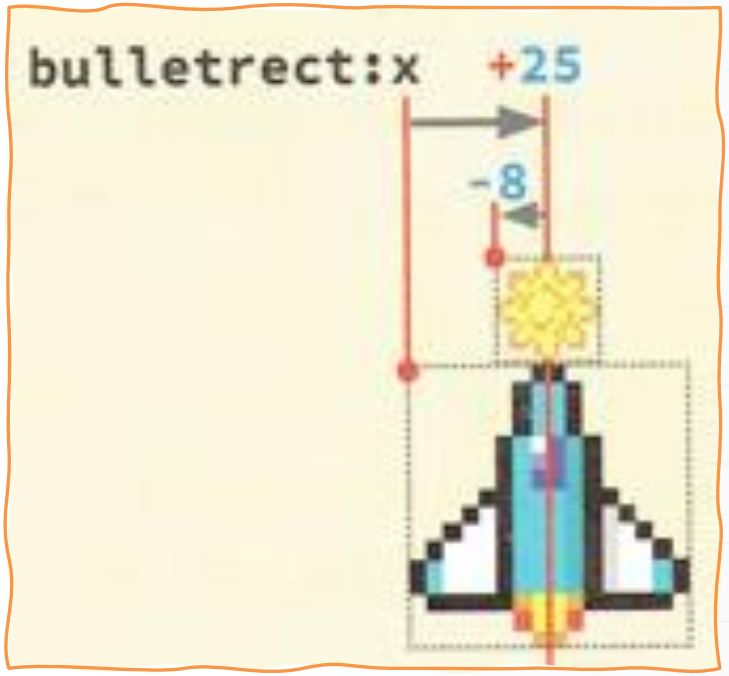
弾が画面の外に出て消えたら次の弾として再登場させる方法で弾を作る。弾を発射する条件は「マウスが押されたか」と「発射できる弾はあるか（弾のY座標が0より小さいか)」の２つ。




```
if mdown[0] and bulletrect.y < 0:
```

解説3:弾の処理について

弾は自機の中心から発射する。「自機のX座標＋25」が自機の中心となる。そこから8（弾の半分の幅）を引いた位置に表示させれば、ちょうど中心に描画される。



bulletrect.x = myrect.x + 25 - 8

復習＆チャレンジ

ここまで習ったことをScratchでもできるかチャレンジしてみよう。
その過程でScratchでできること、Pythonでないとできないことを整理してみよう。

メモ

プログラミング教室のテクノロ

なまえ：