プログラミング教室のテクノロ

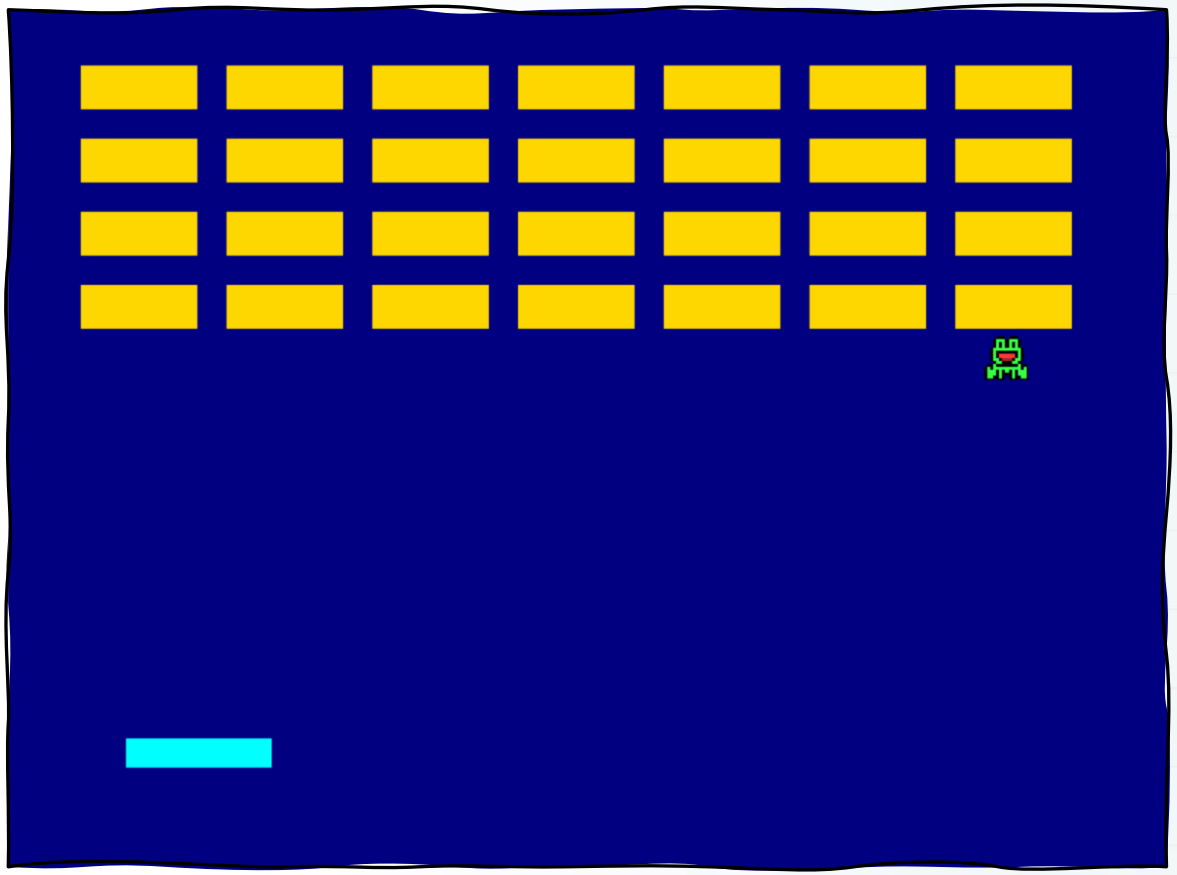
Pythonの道

pygameの使い方⑤

もくじ

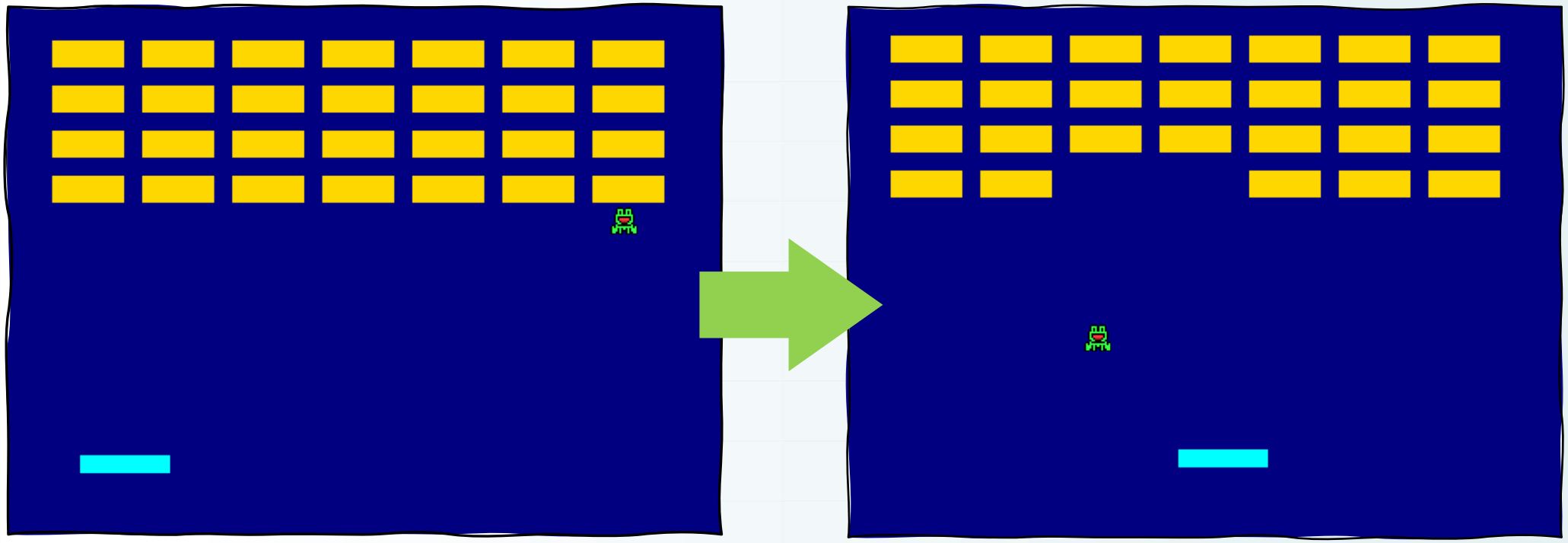
・pygameの使い方⑤

pygameを使って本格的なゲーム開発を学ぶ



ボールをバーで打ち返す

ボールをバーで打ち返して、ブロックを消すゲームを作る。



「動き回るボール」と「反射させるバー」から作る。「バー」は、すばやくコントロールするためマウスで操作する仕様にする。マウスを左右に移動させると、バーがマウスにくっついて左右に移動する。水平に移動させるために、縦方向は固定する。

ボールをバーで打ち返す

```
#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))
#バーデータ
barrect = pg.Rect(400, 500, 100, 20)
#ボールデータ
ballimg = pg.image.load("images/kaeru.png")
ballimg = pg.transform.scale(ballimg, (30, 30))
ballrect = pg.Rect(400, 450, 30, 30)
vx = random.randint(-10,10)
vy = -5
#ゲームステージ
def gamestage():
```

・・バーに当たる前の最初の移動量(VXは-10から10までの数字がランダムで代入される)

次のページに続く

ボールをバーで打ち返す

```
#画面を初期化する
global vx, vy
screen.fill(pg.Color("NAVY"))
#ユーザーからの入力を調べる
(mx, my) = pg.mouse.get_pos()
#絵を描いたり、判定したりする
#バーの処理
barrect.x = mx - 50 ・・・解説1
pg.draw.rect(screen, pg.Color("CYAN"), barrect)
#ボールの処理
if ballrect.y < 0:
    vy = -vy
if ballrect.x < 0 or ballrect.x > 800 - 30:
    vx = -vx
if barrect.colliderect(ballrect): ・・・解説3
```

・・・解説2

次のページに続く

ボールをバーで打ち返す

```
            vx = ((ballrect.x + 15) - (barrect.x + 50)) / 4
            vy = random.randint(-10, -5)
            pg.mixer.Sound("sounds/pon.wav").play()
        ballrect.x = ballrect.x + vx
        ballrect.y = ballrect.y + vy
        screen.blit(ballimg, ballrect)
#この下をずっとループする
while True:
    gamestage()
    #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:
```

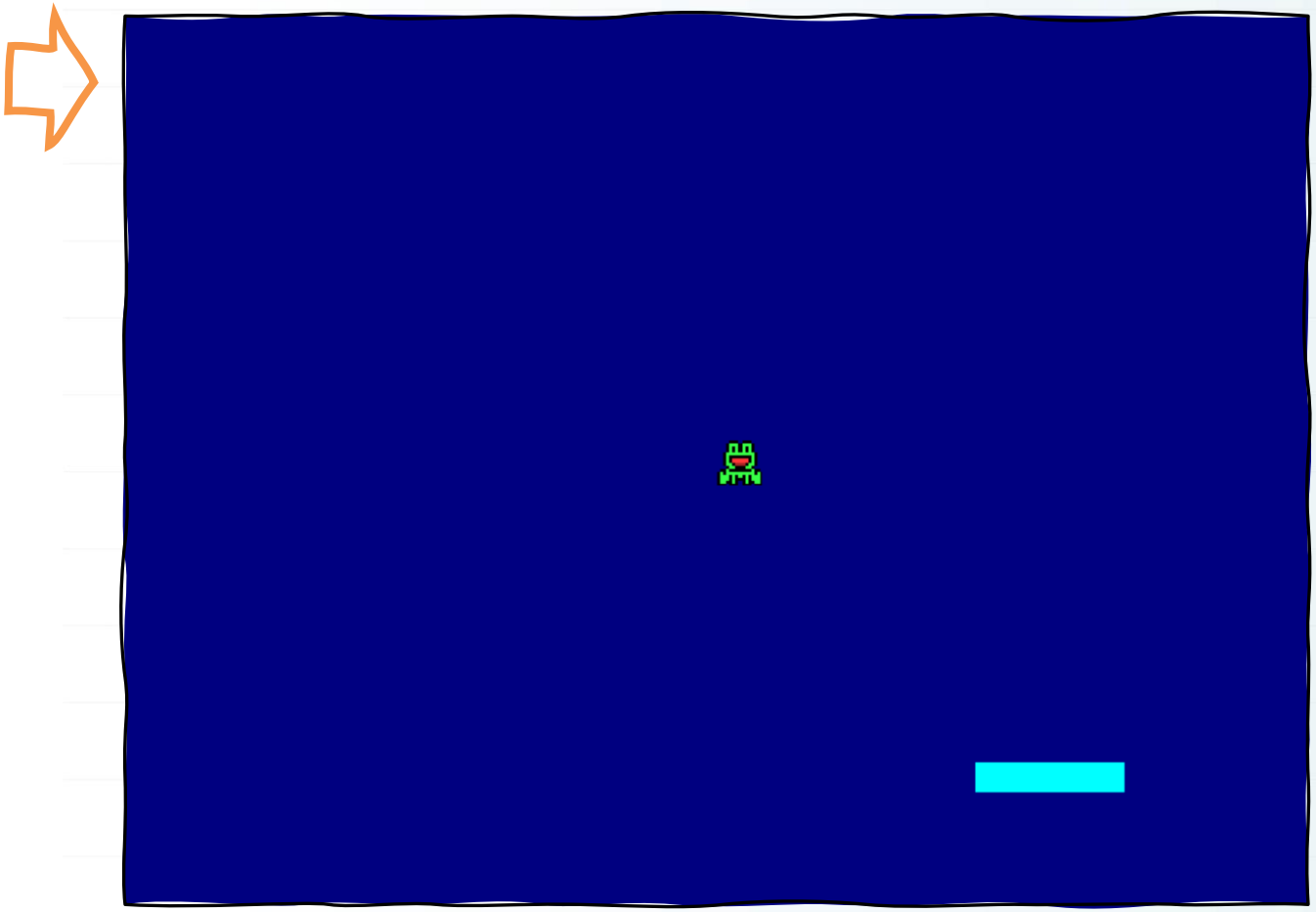
・・・解説3

次のページに続く

ボールをバーで打ち返す

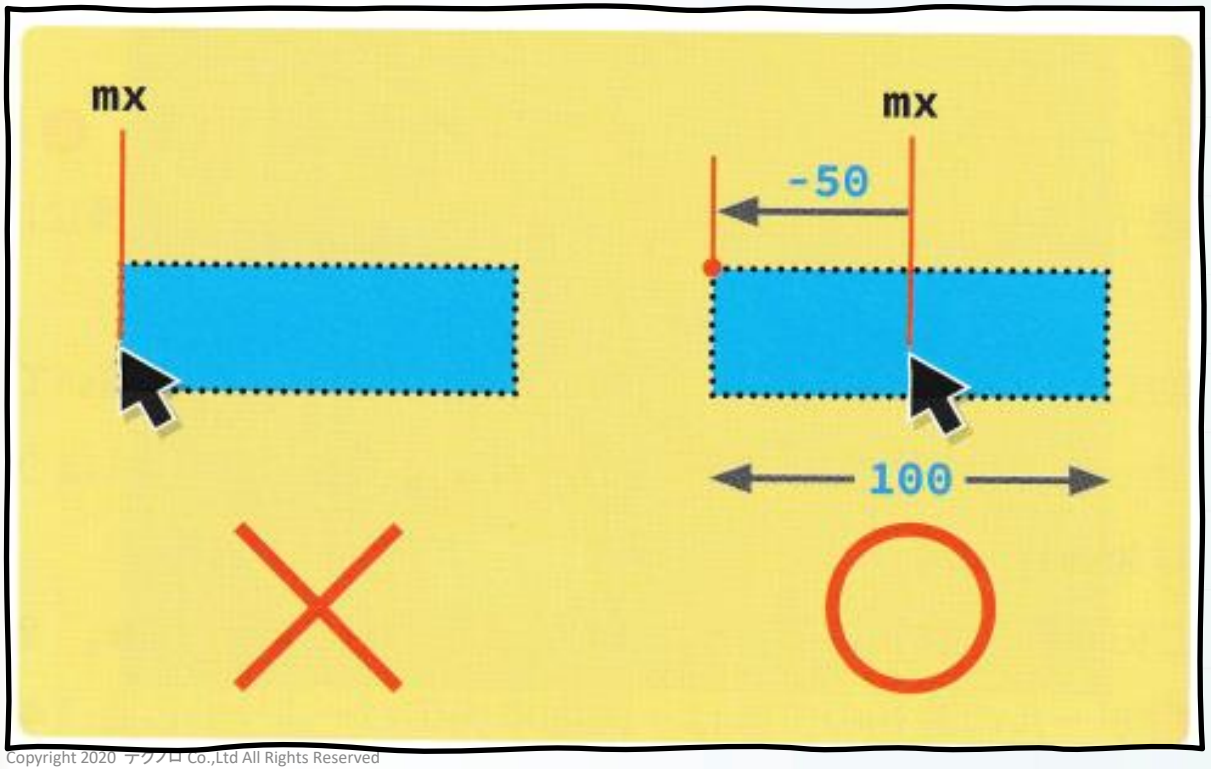
```
pg.quit()
sys.exit()
```

「Run Module」またはF5キーを押してプログラムを実行する。



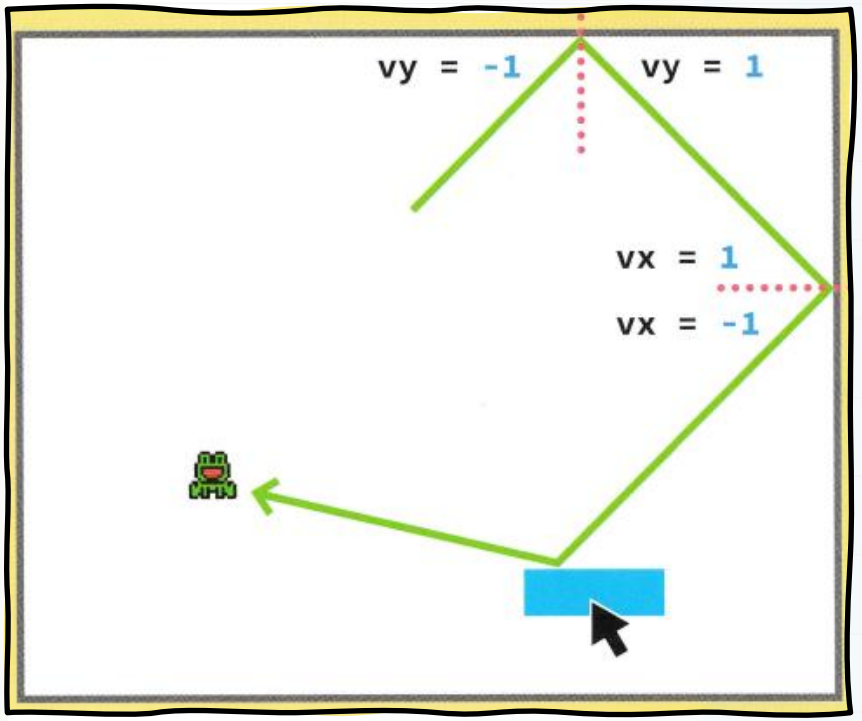
解説1:バーの処理について

(mx,my)=pg.mouse.get_pos()でユーザからの入力を調べた上でマウスの位置が決まる。マウスの位置が決まれば、barrect.x=mx-50でバーを表示する。このとき、取得したマウスのX座標からバーを描き始めると、バーがマウスの右側に描画される。バーの幅が100なので、マウスのX座標から50をひいた位置から描き始めると、マウスがバーの中心になり、つかんで移動させている感じがする。



解説2:ボールの処理について①

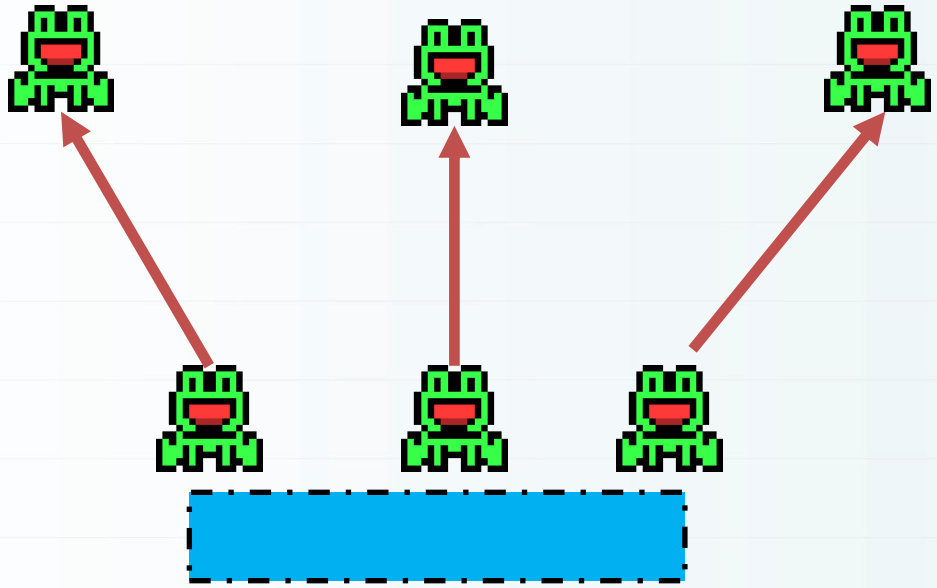
ボールは画面の上や左右から出ないようにする。ボールのY座標が0より小さければ、ボールが画面の上に出ようとしている。そのときはvyをマイナスにして下に反射させる。もしX座標が0より小さいか、800-ボールの大きさ(30)より大きければ、画面の左右から出ようとしているので、vxをマイナスにして内側に反射させる。




```
if ballrect.y < 0:
        vy = -vy
    if ballrect.x < 0 or ballrect.x > 800 - 30:
        vx = -vx
```

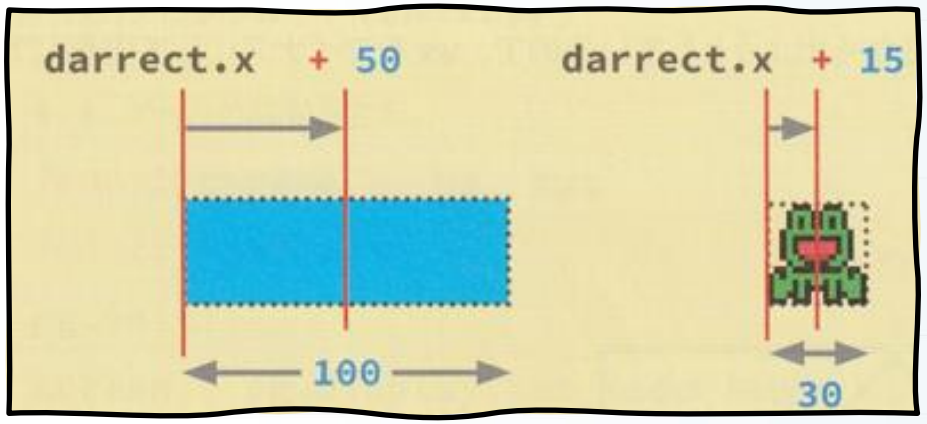
解説3:ボールの処理について②

次にボールとバーの衝突を調べる。衝突の際、「同じ角度で反射するだけ」だと、単純で退屈なゲームになる。そこで「バーのどこに衝突したかで反射の向きが変わる」仕組みを実装する。

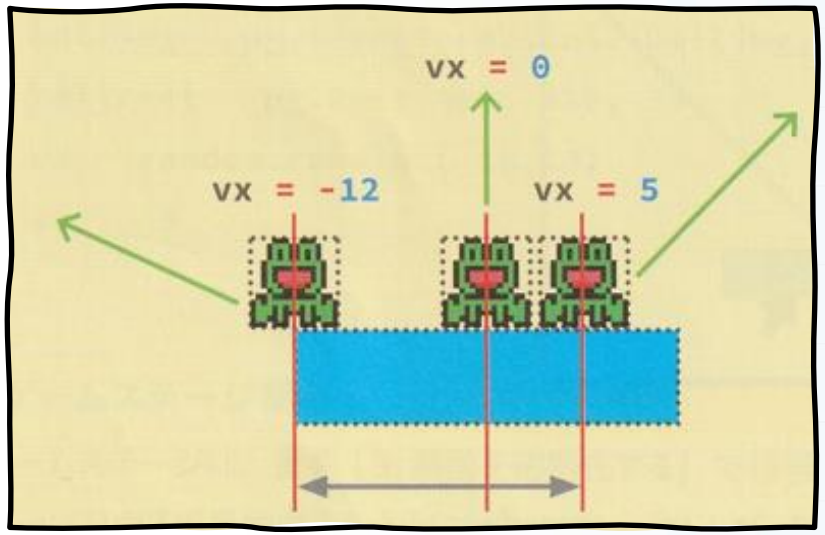


衝突した場所が、バーの真ん中だったら真上(vx=0)に反射し、右側だったら右方向(vx=マイナス)、左側だったら左方向(vx=マイナス)に反射するようにすれば、ボールの向きをコントロールすることができる。さらに「真ん中より離れる程、より強く左右へ反射する」ようにすれば、練習して上達したくなるゲームになる。

解説3:ボールの処理について②



「どこに衝突したか」はボールとバーの中心のX座標で調べることができる。バーは幅100なので「X座標に50足した位置」、ボールは幅30なので「X座標に15足した位置」がそれぞれのX座標となる。



「ボールのX座標－バーのX座標」でボールがバーの右に衝突するほどプラス、バーの左に衝突するほどマイナスの値になる。ただし、その値そのままだと極端になりすぎるので4で割って調整している。

解説3:ボールの処理について②

上方向へのスピードは-10~-5でランダムにして、vyにいれる。

```
vy = random.randint(-10, -5)
```

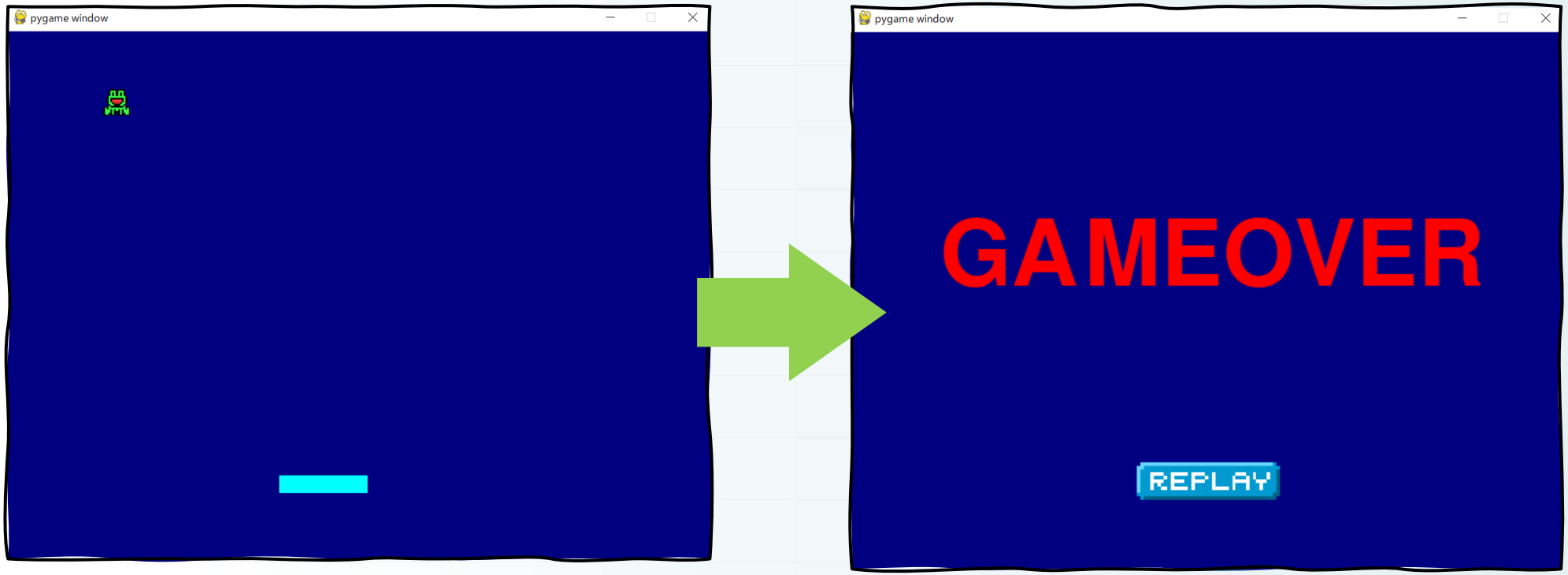
これで、バーのどこに衝突したかで反射の向きが変わるしかけができた。さらに反射したときは「ピッ」というサウンドを鳴らす。

```
pg.mixer.Sound("sounds/pon.wav").play()
```

最後に移動量を足して、ボールを移動させる。

ゲームオーバを作る

ボールが画面の下に移動したら、ゲームオーバにする。画面の下に落ちたらゲームオーバ画面に切り替える。



ボールが落ちたことを調べるので「ゲームステージ関数」を修正する。「ゲームオーバー画面関数」を作って、「メインループ」から切り替えられるようにする。何度も遊べるようにするために「ゲームリセット関数」も用意する。

ゲームオーバを作る

```
#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))
#バーデータ
barrect = pg.Rect(400, 500, 100, 20)
#ボールデータ
ballimg = pg.image.load("images/kaeru.png")
ballimg = pg.transform.scale(ballimg, (30, 30))
ballrect = pg.Rect(400, 450, 30, 30)
vx = random.randint(-10,10)
vy = -5
#ボタンデータ
replay_img = pg.image.load("images/replaybtn.png")
```

・・追加する箇所

次のページに続く

ゲームオーバを作る

```
#メインループで使う変数
pushFlag = False   ・・「ボタンを押したFlag」をFalseにする
page = 1   ・・ページ番号を1に初期化する
#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]:
        if btn.collidepoint(mx, my) and pushFlag == False:
            pg.mixer.Sound("sounds/pi.wav").play()
            page = newpage
            pushFlag = True
    else:
```

・・追加する箇所

・・追加する箇所

・・追加する箇所

次のページに続く

ゲームオーバを作る

```
        pushFlag = False
```

・・追加する箇所

```
#ゲームステージ
def gamestage():
    #画面を初期化する
    global vx, vy
    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()
    #絵を描いたり、判定したりする
    #バーの処理
    barrect.x = mx - 50
    pg.draw.rect(screen, pg.Color("CYAN"), barrect)
    #ボールの処理
    if ballrect.y < 0:
```

次のページに続く

ゲームオーバを作る

```
        vy = -vy
    if ballrect.x < 0 or ballrect.x > 800 - 30:
        vx = -vx
    if barrect.colliderect(ballrect):
        vx = ((ballrect.x + 15) - (barrect.x + 50))  / 4
        vy = random.randint(-10, -5)
        pg.mixer.Sound("sounds/pon.wav").play()
    if ballrect.y > 600:
        page = 2
        pg.mixer.Sound("sounds/down.wav").play()
    ballrect.x = ballrect.x + vx
    ballrect.y = ballrect.y + vy
    screen.blit(ballimg, ballrect)
#データのリセット
def gamereset():
```

・・ボールが画面の下にきたら、ページを２にする

・・追加する箇所

・・追加する箇所

次のページに続く

ゲームオーバを作る

```
    global vx, vy
    vx = random.randint(-10,10)
    vy = -5
    ballrect.x = 400
    ballrect.y = 450
```

・・追加する箇所

```
#ゲームオーバー
def gameover():
    gamereset()
    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))
    #絵を描いたり、判定したりする
    button_to_jump(btn1, 1)
```

・・追加する箇所

次のページに続く

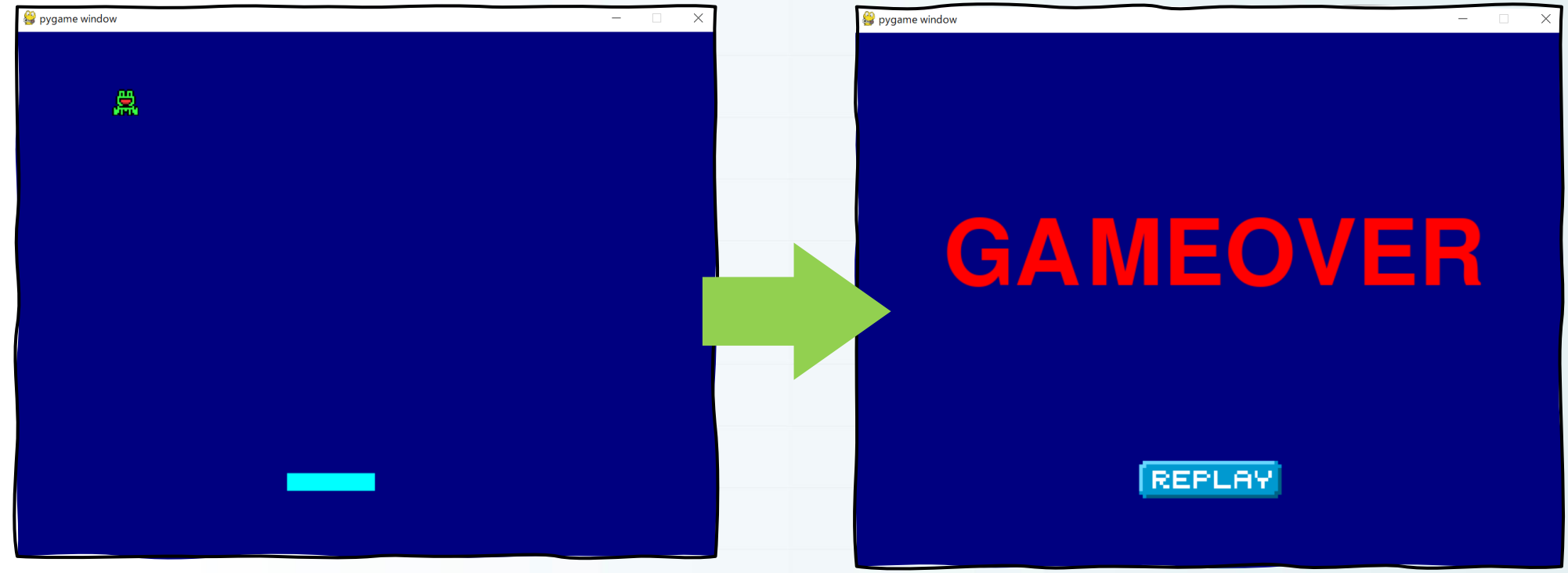
ゲームオーバを作る

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

・・追加する箇所

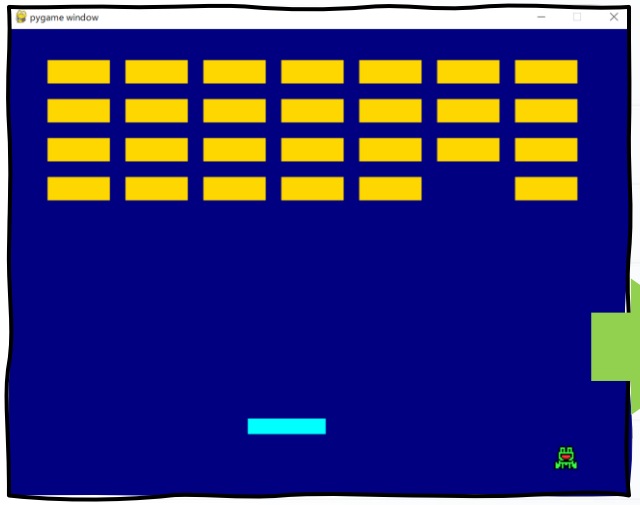
ゲームオーバを作る

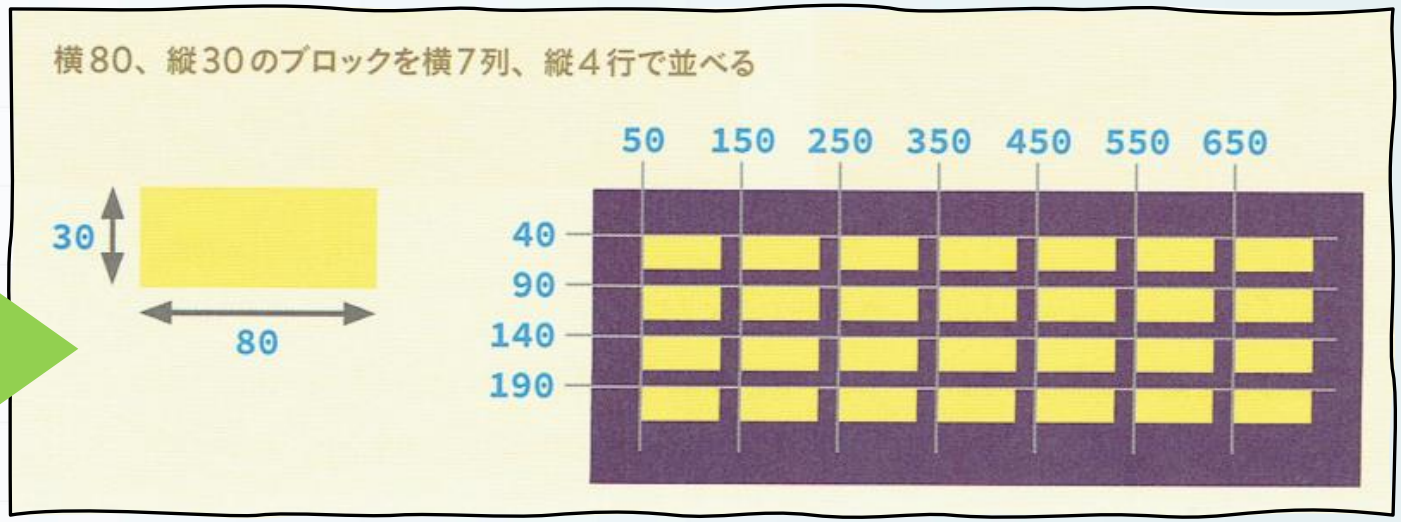
「Run Module」またはF5キーを押してプログラムを実行する。



ブロックをたくさん並べる

ブロックをたくさん作る。「規則正しい並び」はfor文を使って位置を計算して行う。






```
for y in range(4):
    for x in range(7):
        print(" x=" ,50 + x*100, " y= " ,40+y*50)
```

xとyにはfor文の入れ子の中で以下の値が入る

x=50 y=40

x=150 y=40

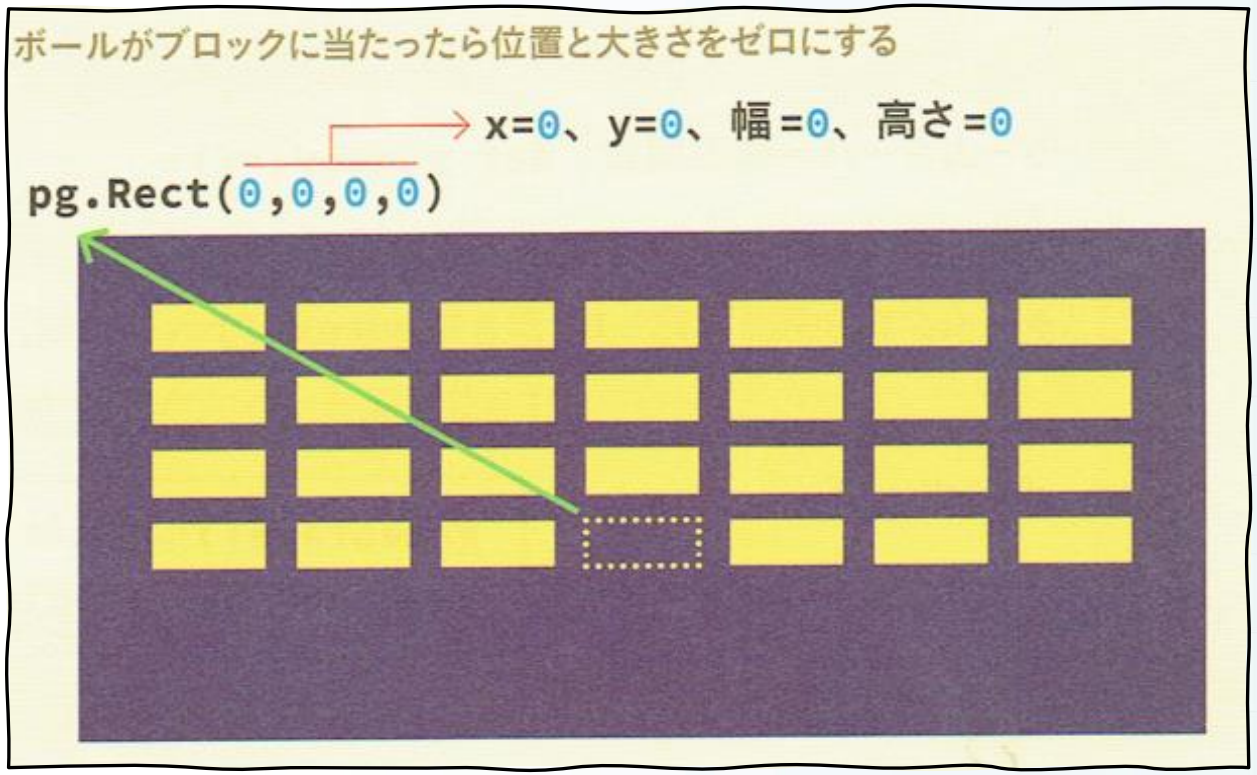
x=250 y=40

:

:

ブロックをたくさん並べる

ボールがブロックと衝突したとき、ブロックを消す機能を作る。
「位置と大きさをゼロ」にしてブロックを消す。



さらに「ブロックを全部消した時の処理」も必要となり、ブロックを消すたびに数を足していき、消した数が28個になったら「ゲームクリア画面」に切り替えるようにする。

ブロックをたくさん並べる

```
#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))
#バーデータ
barrect = pg.Rect(400, 500, 100, 20)
#ボールデータ
ballimg = pg.image.load("images/kaeru.png")
ballimg = pg.transform.scale(ballimg, (30, 30))
ballrect = pg.Rect(400, 450, 30, 30)
vx = random.randint(-10,10)
vy = -5
#ボタンデータ
replay_img = pg.image.load("images/replaybtn.png")
```

次のページに続く

ブロックをたくさん並べる

```
#ブロックデータ
blocks = []・・空のリストを作成
for yy in range(4):
    for xx in range(7):
        blocks.append(pg.Rect(50+xx*100, 40+yy*50, 80, 30))
#メインループで使う変数
pushFlag = False
page = 1
score = 0・・消したブロックの数を数える変数
#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()
```

```
1. a = [1,2,3]
2. a.append([4,5])
3. print(a)
```

・リストに値を追加する関数

・・追加する箇所

・・追加する箇所

次のページに続く

ブロックをたくさん並べる

```
    if mdown[0]:
        if btn.collidepoint(mx, my) and pushFlag == False:
            pg.mixer.Sound("sounds/pi.wav").play()
            page = newpage
            pushFlag = True
    else:
        pushFlag = False
#ゲームステージ
def gamestage():
    #画面を初期化する
    global vx, vy
    global page
    global score
    screen.fill(pg.Color("NAVY"))
    #ユーザーからの入力を調べる
```

・・追加する箇所

次のページに続く

ブロックをたくさん並べる

```
    (mx, my) = pg.mouse.get_pos()
    #絵を描いたり、判定したりする
    #バーの処理
    barrect.x = mx - 50
    pg.draw.rect(screen, pg.Color("CYAN"), barrect)
    #ボールの処理
    if ballrect.y < 0:
        vy = -vy
    if ballrect.x < 0 or ballrect.x > 800 - 30:
        vx = -vx
    if barrect.colliderect(ballrect):
        vx = ((ballrect.x + 15) - (barrect.x + 50))  / 4
        vy = random.randint(-10, -5)
        pg.mixer.Sound("sounds/pon.wav").play()
    if ballrect.y > 600:
```

次のページに続く

ブロックをたくさん並べる

```
        page = 2
        pg.mixer.Sound("sounds/down.wav").play()
    ballrect.x = ballrect.x + vx
    ballrect.y = ballrect.y + vy
    screen.blit(ballimg, ballrect)
    #ブロックの処理
    n = 0
    for block in blocks:
        pg.draw.rect(screen, pg.Color("GOLD"), block)
        #ブロックとボールが衝突したら、ボールを跳ね返してブロックを消す
        if block.colliderect(ballrect):
            pg.mixer.Sound("sounds/piko.wav").play()
            vy = -vy
            blocks[n] = pg.Rect(0,0,0,0)
            score = score + 1
```

・・追加する箇所

次のページに続く

ブロックをたくさん並べる

```
            if score == 28:
                pg.mixer.Sound("sounds/up.wav").play()
                page = 3
        n = n + 1
#データのリセット
def gamereset():
    global vx, vy
    global score
    global blocks
    vx = random.randint(-10, 10)
    vy = -5
    ballrect.x = 400
    ballrect.y = 450
    score = 0
    blocks = []
```

・・追加する箇所

・・追加する箇所

・・追加する箇所

次のページに続く

ブロックをたくさん並べる

```
    for yy in range(4):
        for xx in range(7):
            blocks.append(pg.Rect(xx*100+50, yy*50+40, 80, 30))
#ゲームオーバー
def gameover():
    gamereset()
    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))
    #絵を描いたり、判定したりする
    button_to_jump(btn1, 1)
#ゲームクリア
def gameclear():
```

・・追加する箇所

・・追加する箇所

次のページに続く

ブロックをたくさん並べる

```
    gamereset()
    screen.fill(pg.Color("GOLD"))
    font = pg.font.Font(None, 150)
    text = font.render("GAMECLEAR", True, pg.Color("RED"))
    screen.blit(text, (60, 200))
    btn1 = screen.blit(replay_img,(320, 480))
    #絵を描いたり、判定したりする
    button_to_jump(btn1, 1)
#この下をずっとループする
while True:
    if page == 1:
        gamestage()
    elif page == 2:
        gameover()
    elif page == 3:
```

・・追加する箇所

・文字列の画像を作る関数

・・・page変数を見て、gamestage()か、gameover()か、gameclear()かを切り替える

・・追加する箇所

次のページに続く

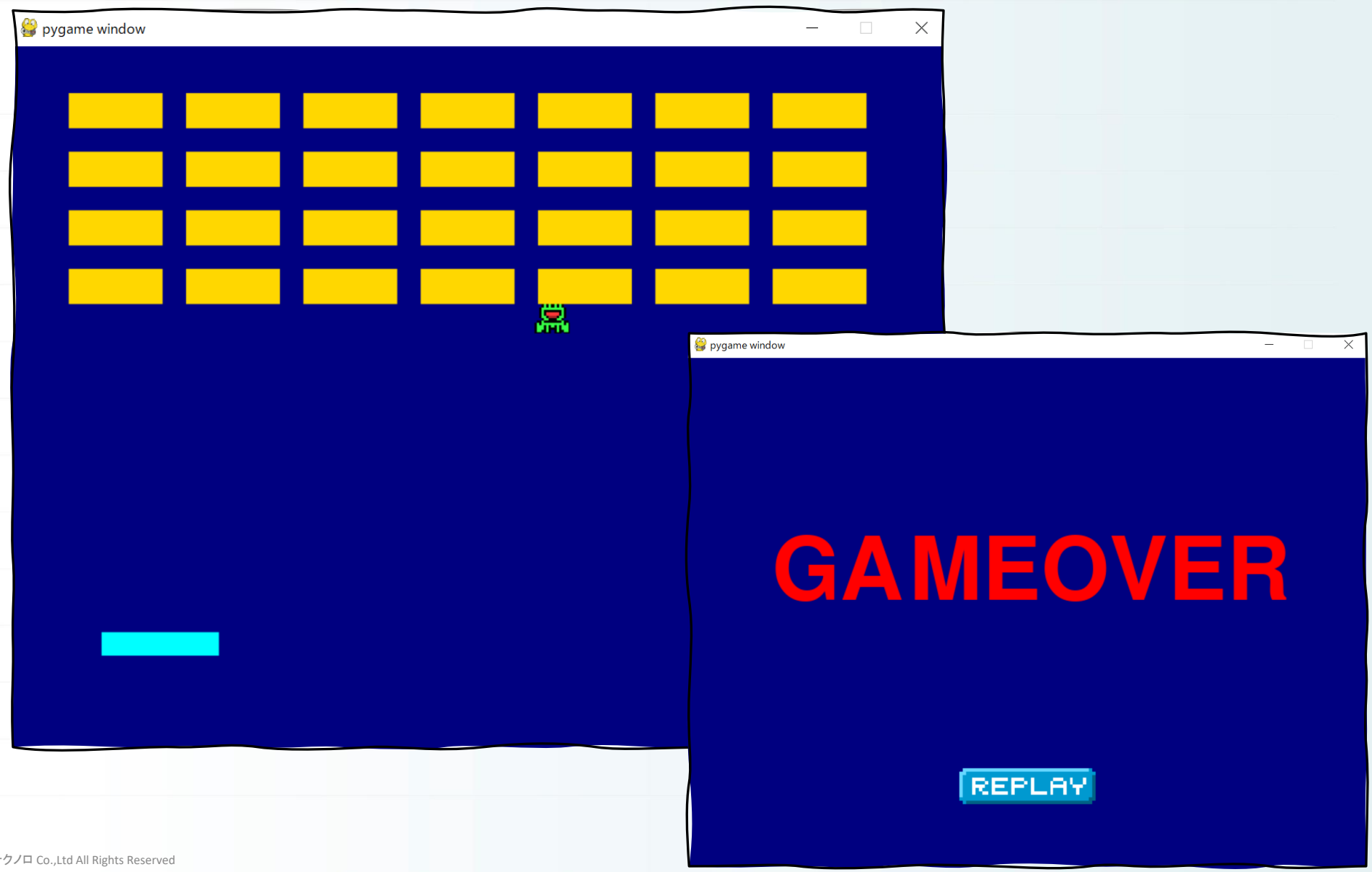
ブロックをたくさん並べる

```
    gameclear()
#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:
        pg.quit()
        sys.exit()
```

・・追加する箇所

ボールをバーで打ち返す

「Run Module」またはF5キーを押してプログラムを実行する。



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：