プログラミング教室のテクノロ

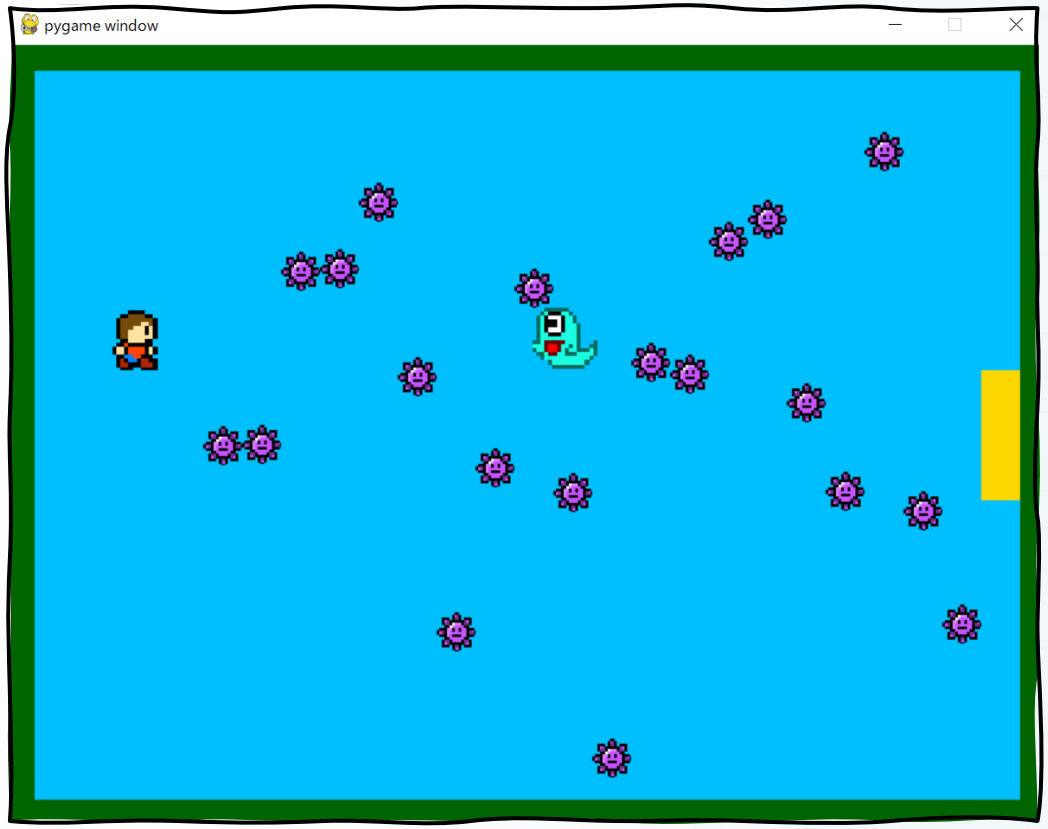
Pythonの道

pygameの使い方④(後半)

もくじ

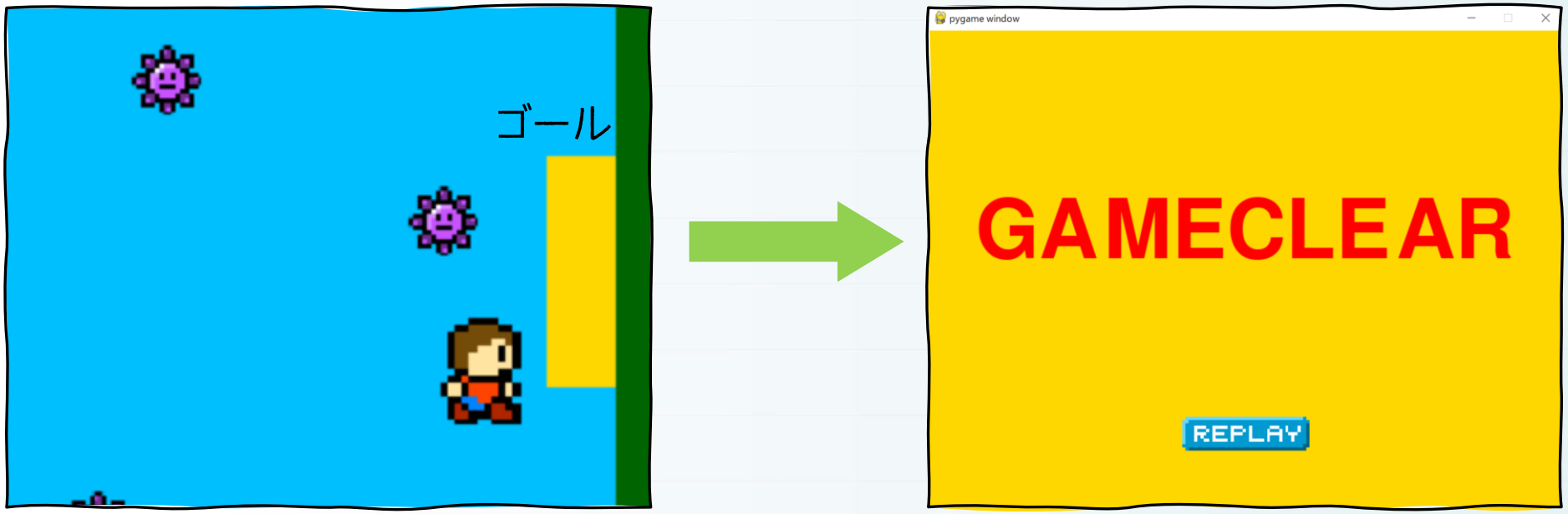
・pygameの使い方④

pygameを使って本格的なゲーム開発を学ぶ



ゴールと衝突したらゲームクリア

画面の右端にゴールを作り、プレイヤがゴールに触れたらゲームクリアになるようにする。



「ゲームの準備」でゴールのデータを作り、「ゲームステージ関数」にゴールの処理を追加する。

ゴールと衝突したらゲームクリア

```
#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))・・ゲーム用ウィンドウを作る
#プレイヤデータ
myimgR = pg.image.load("images/playerR.png")
myimgR = pg.transform.scale(myimgR, (40, 50)) ・・画像サイズの変更
myimgL = pg.transform.flip(myimgR, True, False) ・・回転方向を左右のみ
myrect = pg.Rect(50,200,40,50) ・・プレイヤの初期位置と大きさを定義
#壁データ
walls = [pg.Rect(0,0,800,20),    ・・４つの壁の位置と大きさをリストにいれる
         pg.Rect(0,0,20,600),
         pg.Rect(780,0,20,600),
         pg.Rect(0,580,800,20)]
```

次のページに続く

ゴールと衝突したらゲームクリア

```
#ワナデータ
trapimg = pg.image.load("images/uni.png")
trapimg = pg.transform.scale(trapimg, (30, 30))
traps = []
for i in range(20):
    wx = 150 + i * 30
    wy = random.randint(20,550)
    traps.append(pg.Rect(wx,wy,30,30))
#ボタンデータ
replay_img = pg.image.load("images/replaybtn.png")
goalrect = pg.Rect(750,250,30,100)
#メインループで使う変数
rightFlag = True
pushFlag = False
page = 1
```

・・追加する箇所

次のページに続く

ゴールと衝突したらゲームクリア

```
#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:
        pushFlag = False
#ゲームステージ
def gamestage():
```

次のページに続く

ゴールと衝突したらゲームクリア

```
#画面を初期化する
global rightFlag
global page
screen.fill(pg.Color("DEEPSKYBLUE"))
vx = 0
vy = 0
#ユーザからの入力を調べる
key = pg.key.get_pressed()
#絵を描いたり、判定したりする
if key[pg.K_RIGHT]:
    vx = 4
    rightFlag = True
if key[pg.K_LEFT]:
    vx = -4
    rightFlag = False
```

次のページに続く

ゴールと衝突したらゲームクリア

```
if key[pg.K_UP]:
    vy = -4
if key[pg.K_DOWN]:
    vy = 4
#プレイヤの処理
myrect.x = myrect.x + vx
myrect.y = myrect.y + vy
if myrect.collidelist(walls) != -1:
    myrect.x = myrect.x - vx
    myrect.y = myrect.y - vy
if rightFlag:
    screen.blit(myimgR, myrect)
else :
    screen.blit(myimgL, myrect)
#壁の処理
```

次のページに続く

ゴールと衝突したらゲームクリア

```
    for wall in walls:
        pg.draw.rect(screen, pg.Color("DARKGREEN"), wall)
    #ワナの処理
    for trap in traps:
        screen.blit(trapimg, trap)
    if myrect.collidelist(traps) != -1:
        pg.mixer.Sound("sounds/down.wav").play()
        page = 2
    #ゴールの処理
    pg.draw.rect(screen,pg.Color("GOLD"),goalrect)
    if myrect.colliderect(goalrect):
        pg.mixer.Sound("sounds/up.wav").play()
        page = 3
#データのリセット
def gamereset() :
```

・・追加する箇所

次のページに続く

ゴールと衝突したらゲームクリア

```
myrect.x = 50
    myrect.y = 100
    for d in range(20):
        traps[d].x = 150 + d * 30
        traps[d].y = random.randint(20,550)
#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)
```

次のページに続く

ゴールと衝突したらゲームクリア

```
#ゲームクリア
def gameclear():
    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))
    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()
```

・・追加する箇所

次のページに続く

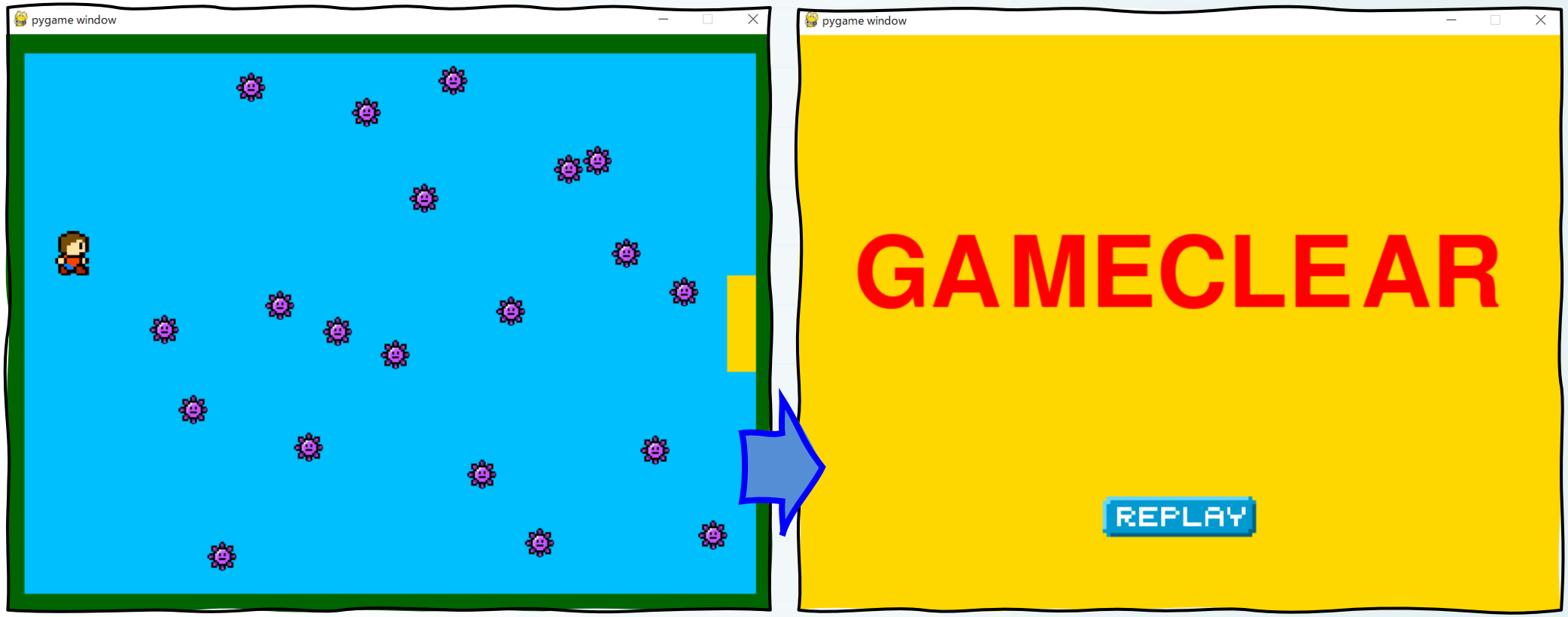
ゴールと衝突したらゲームクリア

```
    elif page == 3:
        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

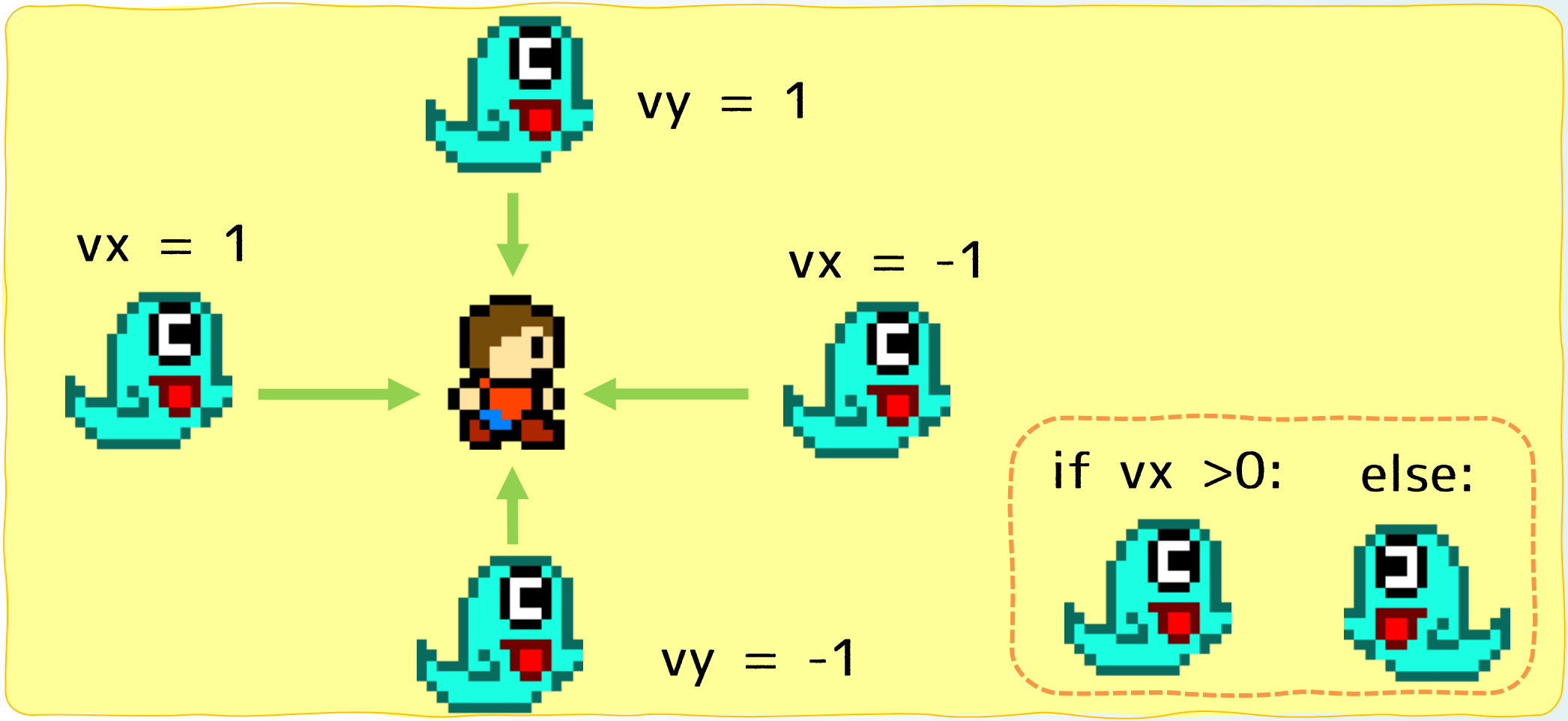
ゴールと衝突したらゲームクリア

「Run Module」またはF5キーを押してプログラムを実行する。



追いかけてくるオバケを追加する

「追いかけてくるオバケ」を登場させて、ゲームを難しくする。プレイヤの位置とオバケの位置を比べて、オバケよりプレイヤが右にいたらオバケを右に、そうでなければオバケを左に移動させる。上下も同じように移動させる。



追いかけてくるオバケを追加する

```
#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))・・ゲーム用ウィンドウを作る
#プレイヤデータ
myimgR = pg.image.load("images/playerR.png")
myimgR = pg.transform.scale(myimgR, (40, 50)) ・・画像サイズの変更
myimgL = pg.transform.flip(myimgR, True, False) ・・回転方向を左右のみ
myrect = pg.Rect(50,200,40,50) ・・プレイヤの初期位置と大きさを定義
#壁データ
walls = [pg.Rect(0,0,800,20),    ・・４つの壁の位置と大きさをリストにいれる
         pg.Rect(0,0,20,600),
         pg.Rect(780,0,20,600),
         pg.Rect(0,580,800,20)]
```

次のページに続く

追いかけてくるオバケを追加する

```
#ワナデータ
trapimg = pg.image.load("images/uni.png")
trapimg = pg.transform.scale(trapimg, (30, 30))
traps = []
for i in range(20):
    wx = 150 + i * 30
    wy = random.randint(20,550)
    traps.append(pg.Rect(wx,wy,30,30))
#ボタンデータ
replay_img = pg.image.load("images/replaybtn.png")
goalrect = pg.Rect(750,250,30,100)
#オバケデータ
enemyimgR = pg.image.load("images/obake.png")
enemyimgR = pg.transform.scale(enemyimgR,(50,50))
enemyimgL = pg.transform.flip(enemyimgR,True,False)
```

・・追加する箇所

次のページに続く

追いかけてくるオバケを追加する

```
enemyrect = pg.Rect(650,200,50,50)
#メインループで使う変数
rightFlag = True
pushFlag = False
page = 1
#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
```

・・追加する箇所

次のページに続く

追いかけてくるオバケを追加する

```
                pushFlag = True
    else:
        pushFlag = False
#ゲームステージ
def gamestage():
    #画面を初期化する
    global rightFlag
    global page
    screen.fill(pg.Color("DEEPSKYBLUE"))
    vx = 0
    vy = 0
    #ユーザからの入力を調べる
    key = pg.key.get_pressed()
    #絵を描いたり、判定したりする
    if key[pg.K_RIGHT]:
```

次のページに続く

追いかけてくるオバケを追加する

```
                pushFlag = True
    else:
        pushFlag = False
#ゲームステージ
def gamestage():
    #画面を初期化する
    global rightFlag
    global page
    screen.fill(pg.Color("DEEPSKYBLUE"))
    vx = 0
    vy = 0
    #ユーザからの入力を調べる
    key = pg.key.get_pressed()
    #絵を描いたり、判定したりする
    if key[pg.K_RIGHT]:
```

次のページに続く

追いかけてくるオバケを追加する

```
        vx = 4
        rightFlag = True
    if key[pg.K_LEFT]:
        vx = -4
        rightFlag = False
    if key[pg.K_UP]:
        vy = -4
    if key[pg.K_DOWN]:
        vy = 4
    #プレイヤの処理
    myrect.x = myrect.x + vx
    myrect.y = myrect.y + vy
    if myrect.collidelist(walls) != -1:
        myrect.x = myrect.x - vx
        myrect.y = myrect.y - vy
```

次のページに続く

追いかけてくるオバケを追加する

```
    if rightFlag:
        screen.blit(myimgR, myrect)
    else :
        screen.blit(myimgL, myrect)
    #壁の処理
    for wall in walls:
        pg.draw.rect(screen, pg.Color("DARKGREEN"), wall)
    #ワナの処理
    for trap in traps:
        screen.blit(trapimg, trap)
    if myrect.collidelist(traps) != -1:
        pg.mixer.Sound("sounds/down.wav").play()
        page = 2
    #ゴールの処理
    pg.draw.rect(screen,pg.Color("GOLD"),goalrect)
```

次のページに続く

追いかけてくるオバケを追加する

```
    if myrect.colliderect(goalrect):
        pg.mixer.Sound("sounds/up.wav").play()
        page = 3
    #オバケの処理
    ovx = 0
    ovy = 0
    if enemyrect.x < myrect.x:
        ovx = 1
    else:
        ovx = -1
    if enemyrect.y < myrect.y:
        ovy = 1
    else:
        ovy = -1
    enemyrect.x = enemyrect.x + ovx
```

・・・追加する箇所

オバケの移動量ovx,ovyを用意する。

オバケの位置とプレイヤの位置を比較して、上下左右のどちらに進むかを決めて移動する。

ovxが0より大きく右へ向かっているときは右向き、そうでなければ左向きのオバケを表示する

次のページに続く

追いかけてくるオバケを追加する

```
    enemyrect.y = enemyrect.y + ovy
    if ovx > 0:
        screen.blit(enemyimgR,enemyrect)
    else:
        screen.blit(enemyimgL,enemyrect)
    if myrect.colliderect(enemyrect):
        pg.mixer.Sound("sounds/down.wav").play()
        page = 2
#データのリセット
def gamereset() :
    myrect.x = 50
    myrect.y = 100
    for d in range(20):
        traps[d].x = 150 + d * 30
        traps[d].y = random.randint(20,550)
```

・・追加する箇所

次のページに続く

追いかけてくるオバケを追加する

```
    enemyrect.x = 650
    enemyrect.y = 200
```

・・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)
#ゲームクリア
def gameclear():
    gamereset()
```

次のページに続く

追いかけてくるオバケを追加する

```
    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))
    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:
        gameclear()
    #画面を表示する
```

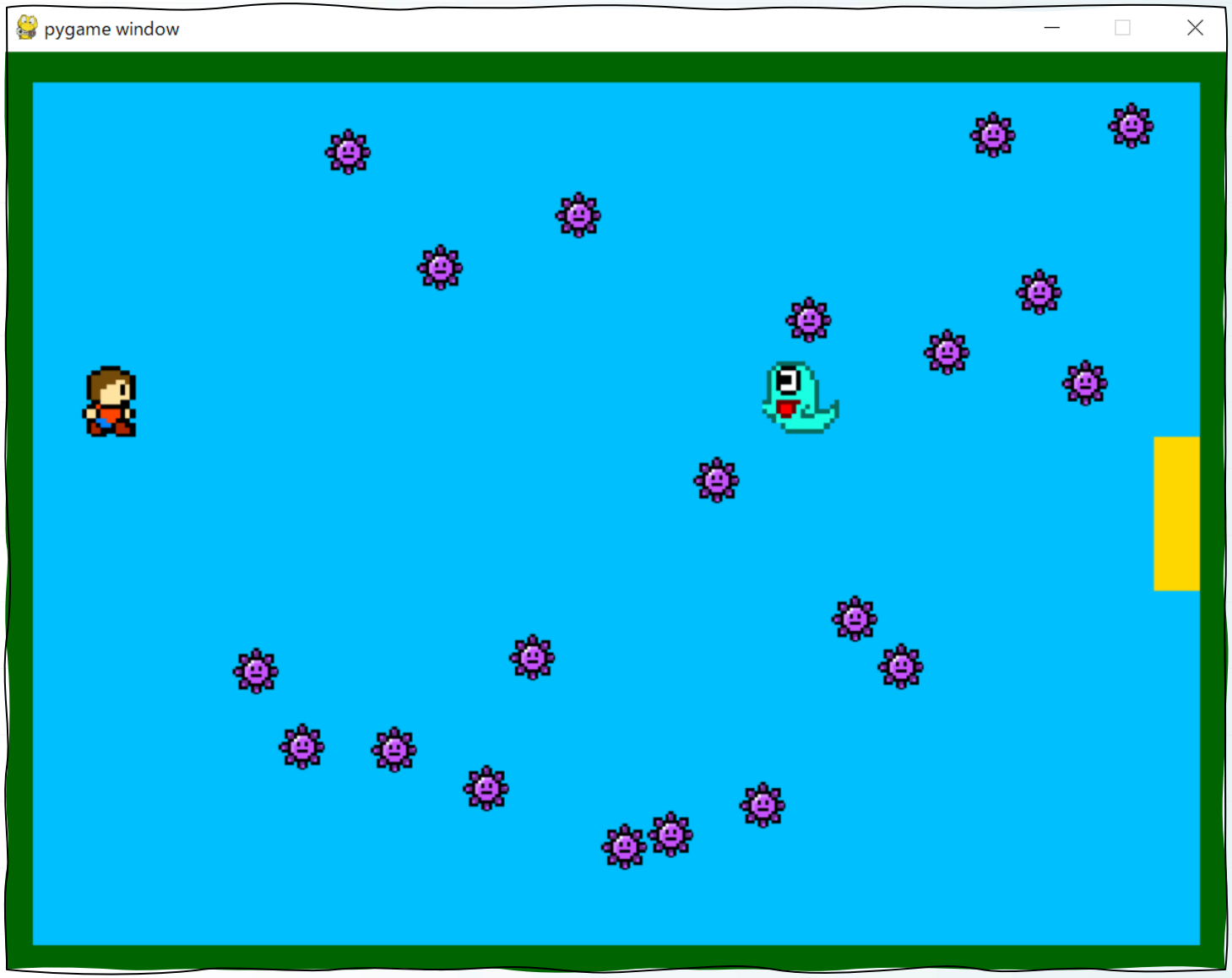
次のページに続く

追いかけてくるオバケを追加する

```
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()
```

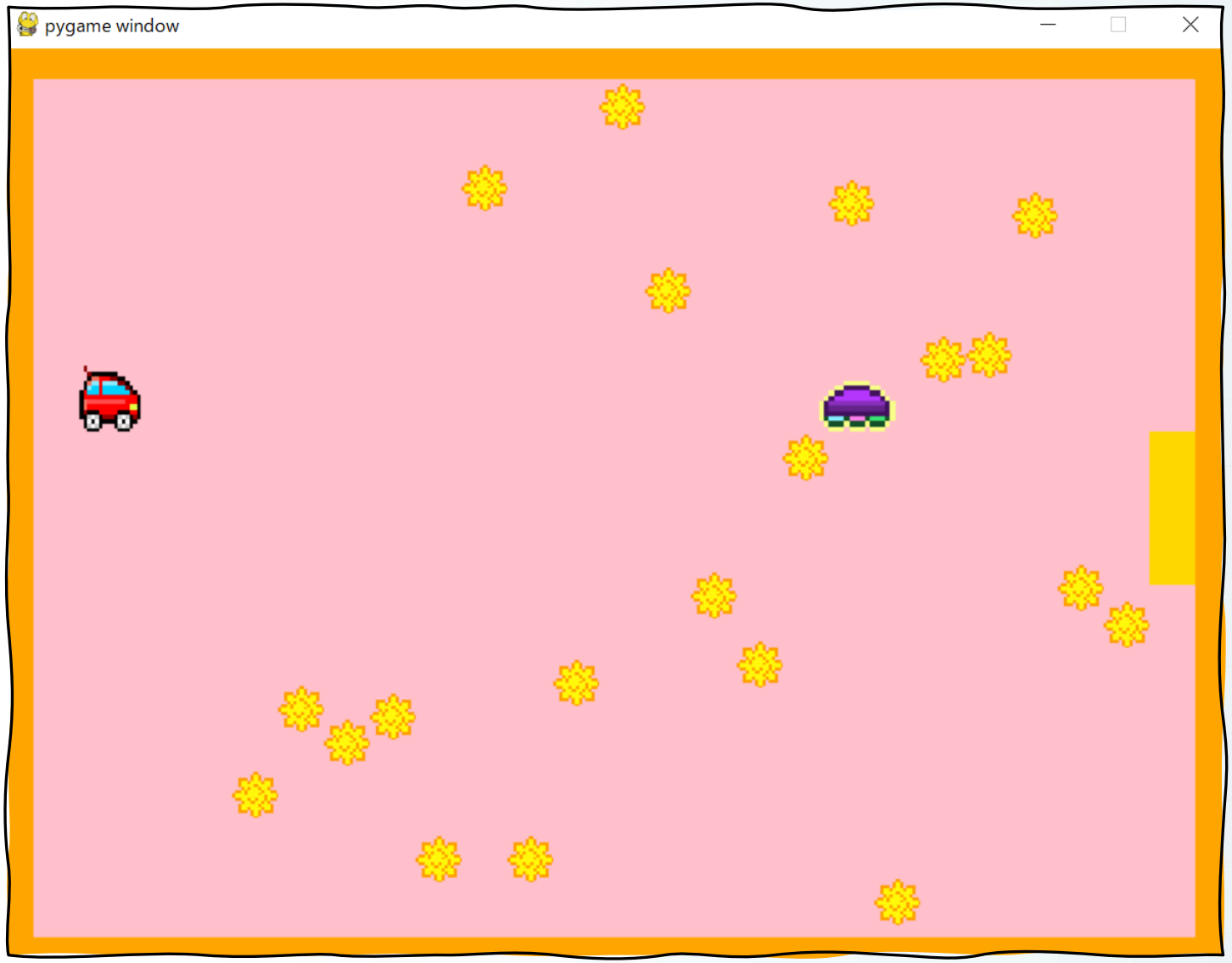
追いかけてくるオバケを追加する

「Run Module」またはF5キーを押してプログラムを実行する。



チャレンジ

自分で簡単なアクションゲームを作ってみよう。



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：