プログラミング教室のテクノロ

Pythonの道

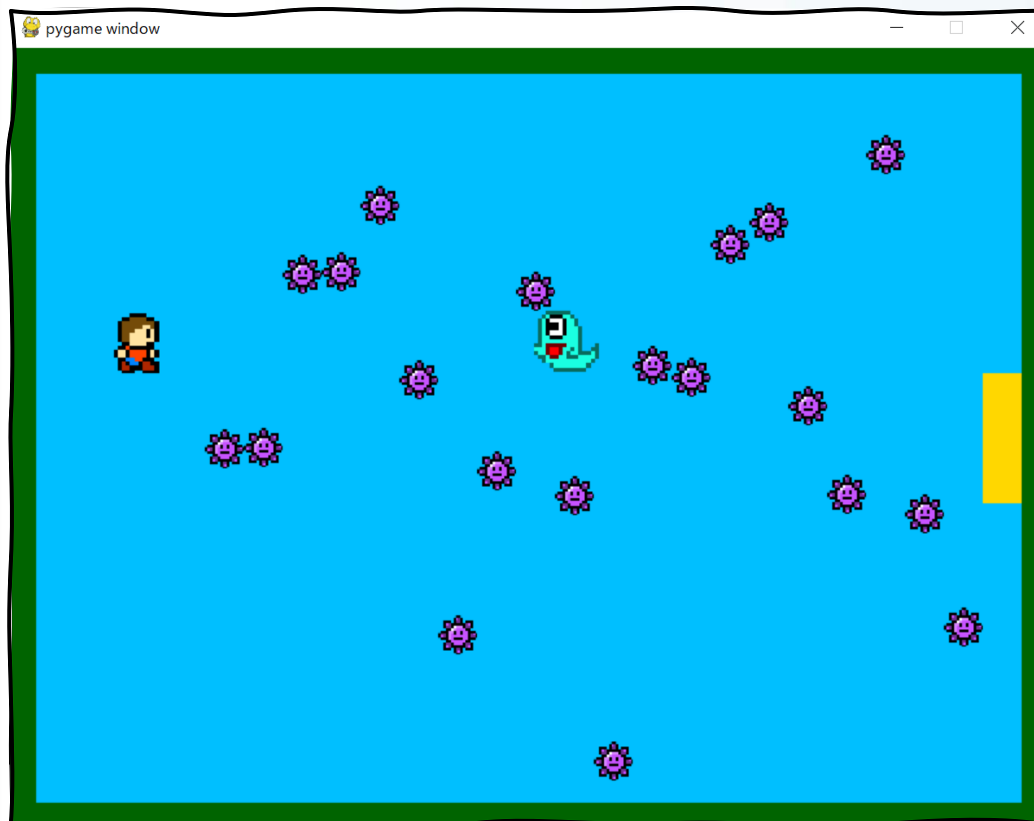
pygameの使い方④(前半)

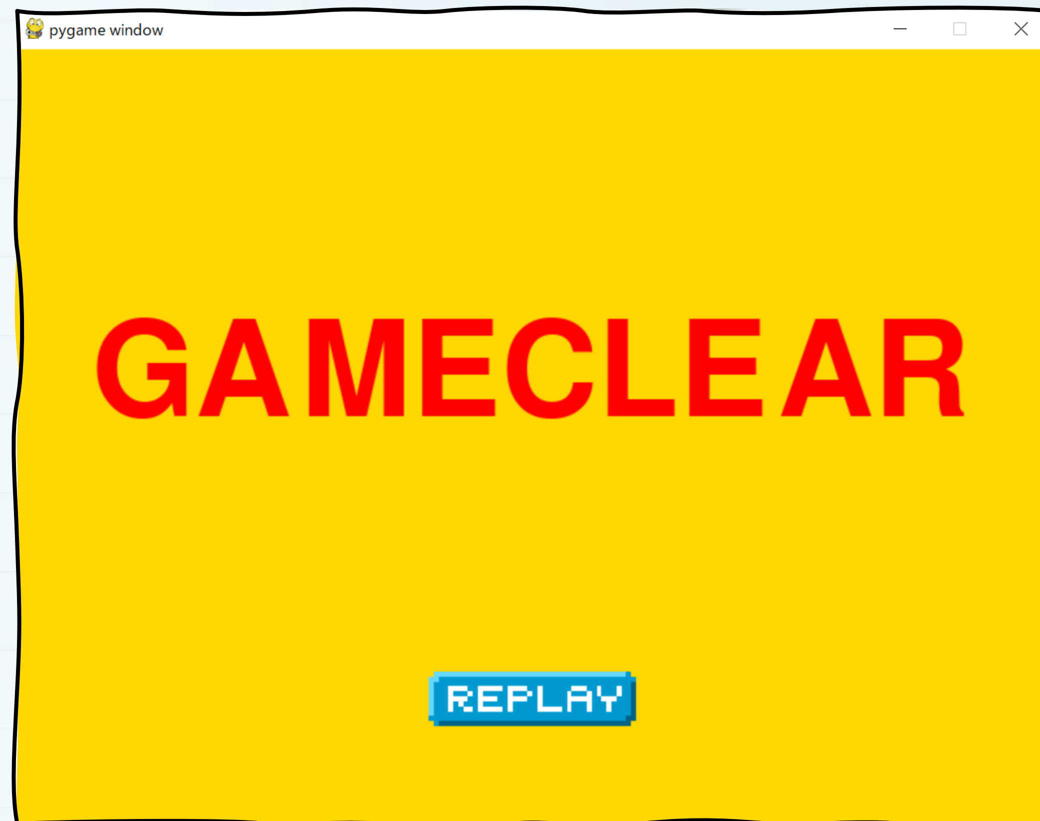
Copyright 2020 テクノロ Co.,LtdAll Rights Reserved

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

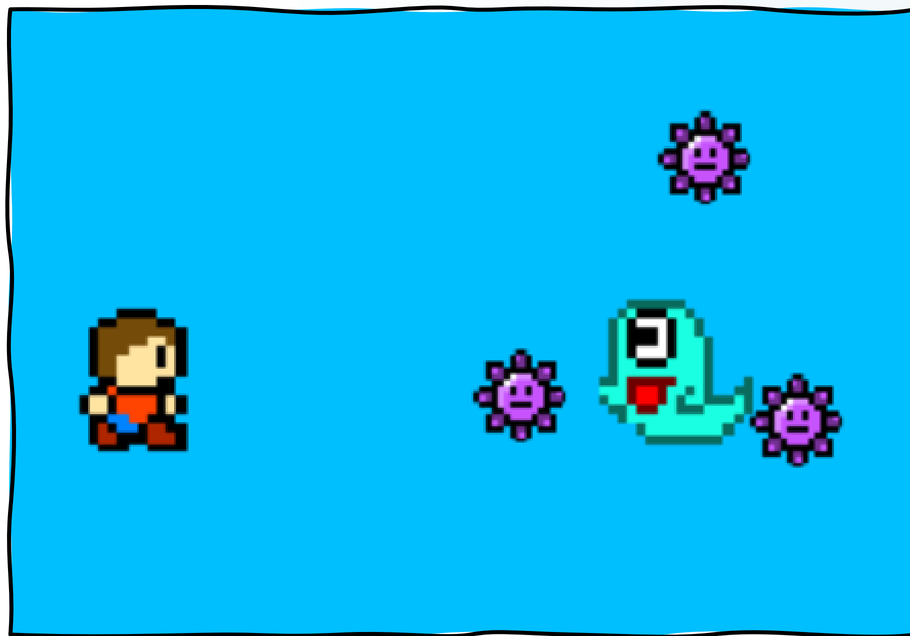


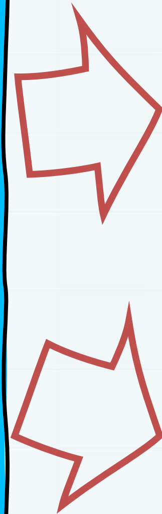


Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

アクションゲームを作る

プレイヤを上下左右に動かして、敵やワナを避けながら、ゴールへたどり着くゲームを作る。

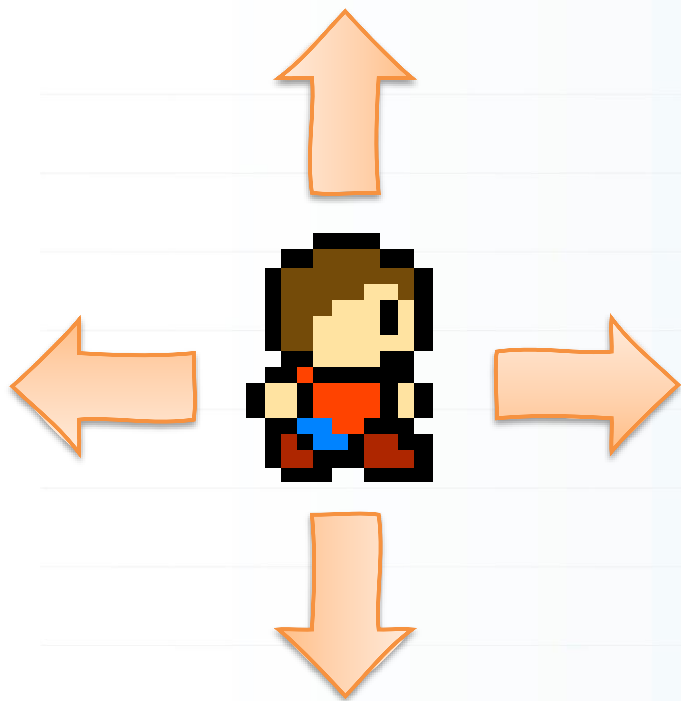


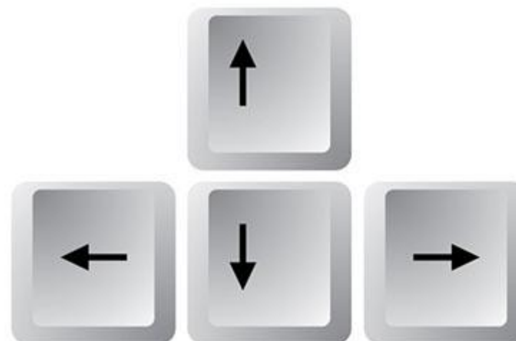


Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

作りたい機能を考える

上下左右キーを押すと、プレイヤが上下左右に動く仕組みを作る。





どのキーが押されているかを調べる

キー変数 = pg.key.get_pressed()

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

キャラクタが動くプログラムを作る

上下左右キーでキャラクタが動き、真ん中に障害物を配置するプログラムを作る。

```
#ゲームの準備をする
import pygame as pg, sys
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)・・画像の左右を反転
myrect = pg.Rect(50,200,40,50) ・・プレイヤの初期位置と大きさを定義
#障害物データ
boxrect = pg.Rect(300,200,100,100)・・障害物の初期位置と大きさを定義
```

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

キャラクタが動くプログラムを作る

```
#メインループで使う変数
rightFlag = True
#ゲームステージ
def gamestage():
    global rightFlag
    #画面を初期化する
    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() ・・今、どのキーがおされているかを調べる
    #絵を描いたり、判定したりする
    if key[pg.K_RIGHT]:  ・・右向き矢印キーが押されたとき
        vx = 4
```

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

キャラクタが動くプログラムを作る

```
        rightFlag = True
    if key[pg.K_LEFT]: ・・左向き矢印キーが押されたとき
        vx = -4
        rightFlag = False
    if key[pg.K_UP]: ・・上向き矢印キーが押されたとき
        vy = -4
    if key[pg.K_DOWN]: ・・下向き矢印キーが押されたとき
        vy = 4
    #プレイヤの処理
    myrect.x = myrect.x + vx   ・・プレイヤのｘ座標を変更
    myrect.y = myrect.y + vy   ・・プレイヤのy座標を変更
    if rightFlag: ・・もし、「rightFlag」がTrueのとき
        screen.blit(myimgR, myrect)
    else:  ・・「rightFlag」がTrueでないとき（False）
```

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

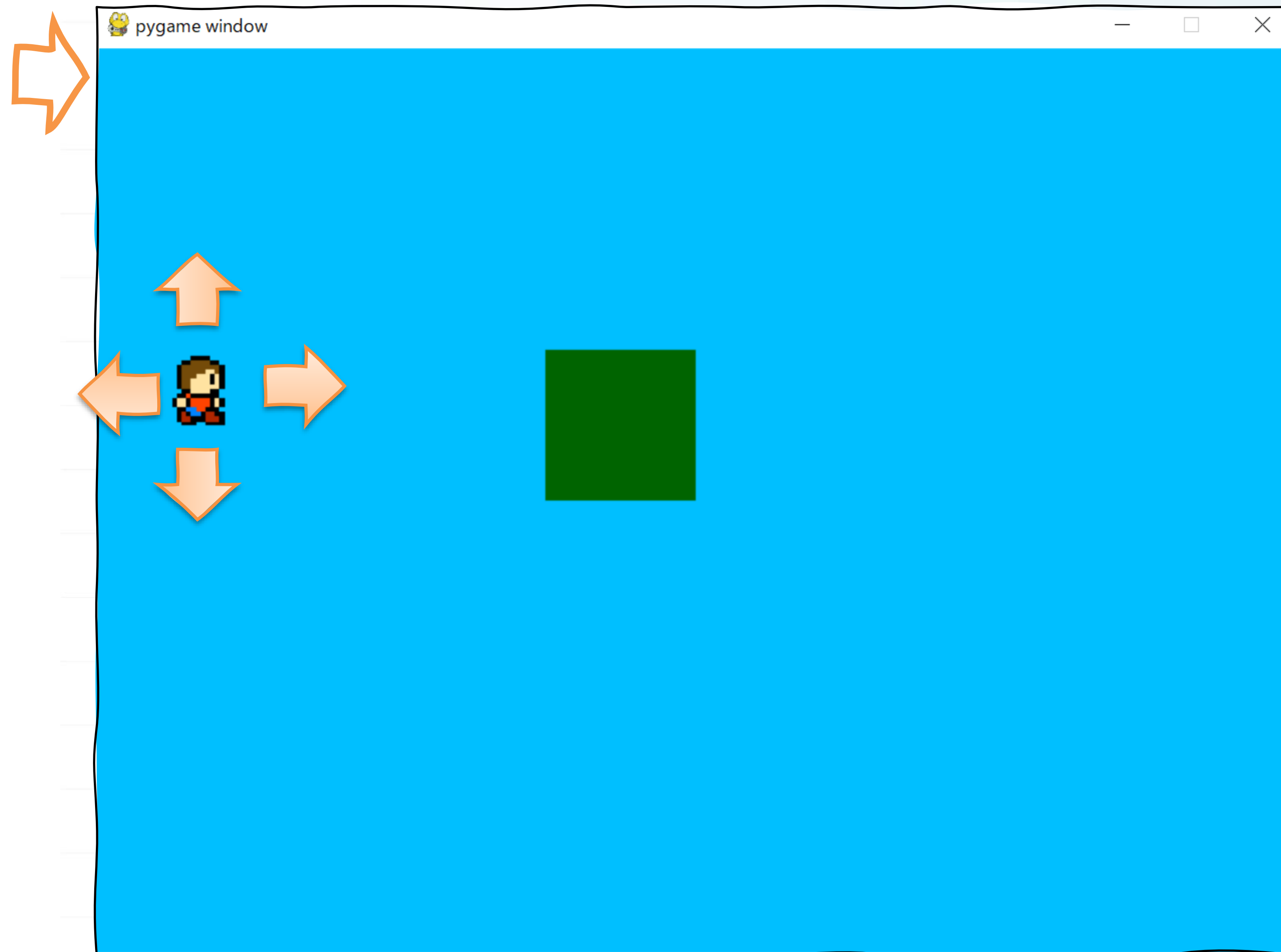
キャラクタが動くプログラムを作る

```
        screen.blit(myimgL, myrect)
    #障害物の処理
    pg.draw.rect(screen, pg.Color("DARKGREEN"), boxrect)
#この下をずっとループする                                ・・四角形を描く
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:
            pg.quit() ・・pygameを終了する
            sys.exit()・・pythonを終了する
```

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

キャラクタが動くプログラムを作る

「Run Module」またはF5キーを押してプログラムを実行する。

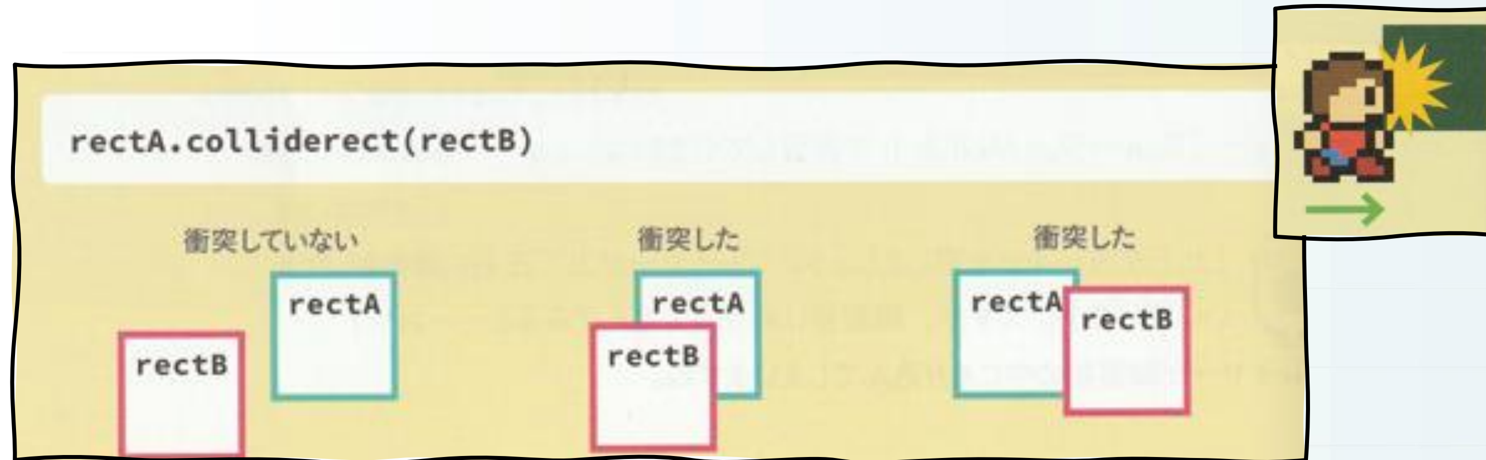


Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

他のRectとの衝突判定

先程のプログラムではプレイヤは障害物にめり込んでしまう。
プレイヤが障害物にめり込まないようにするために２つの絵が
衝突したかを調べる「衝突判定」が必要になる。

衝突判定は「colliderect」関数を使って行う。この関数を使う
と２つの四角形が重なっていないかを調べることができる。
※collide:衝突する



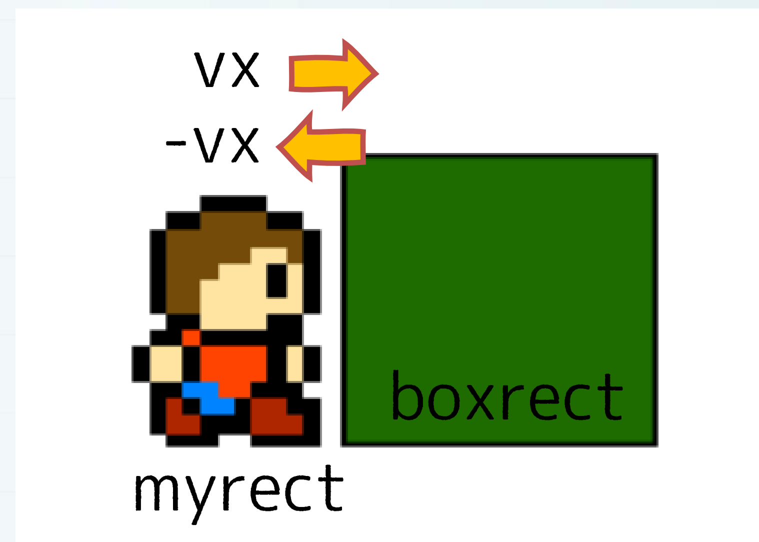
２つのRectが衝突しているかを調べる
変数(衝突したか、していないか) = rectA.colliderect(rectB)

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

他のRectとの衝突判定

プレイヤは「プレイヤの位置にvx、vyを足して移動」している。「もし、vx,vyを足して衝突したなら、vx,vyを引いて戻せば良い。

```
myrect.x = myrect + vx
myrect.y = myrect + vy
if myrect.colliderrect(boxrect):
        myrect.x = myrect – vx
        myrect.y = myrect - vy
```



Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

障害物と衝突するプログラムを作る

真ん中の障害物にあたると衝突するプログラムを作る。

```
#ゲームの準備をする
import pygame as pg, sys
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)・・画像の左右を反転
myrect = pg.Rect(50,200,40,50)・・プレイヤの初期位置と大きさを定義
#障害物データ
boxrect = pg.Rect(300,200,100,100)・・障害物の初期位置と大きさを定義
#メインループで使う変数
```

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

障害物と衝突するプログラムを作る

```
rightFlag = True
#ゲームステージ
def gamestage():
    global rightFlag
    #画面を初期化する
    screen.fill(pg.Color("DEEPSKYBLUE"))
    vx = 0
    vy = 0
    #ユーザー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
```

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

障害物と衝突するプログラムを作る

```
if key[pg.K_LEFT]:
    vx = -4
    rightFlag = False
if key[pg.K_UP]:
    vy = -4
if key[pg.K_DOWN]:
    vy = 4
#プレイヤーの処理
myrect.x = myrect.x + vx
myrect.y = myrect.y + vy
if myrect.colliderect(boxrect):
    myrect.x = myrect.x - vx
    myrect.y = myrect.y - vy
if rightFlag:
```

・・追加する箇所

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

障害物と衝突するプログラムを作る

```
        screen.blit(myimgR, myrect)
    else:
        screen.blit(myimgL, myrect)
    #障害物の処理
    pg.draw.rect(screen, pg.Color("DARKGREEN"), boxrect)
#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:
```

・・追加する箇所

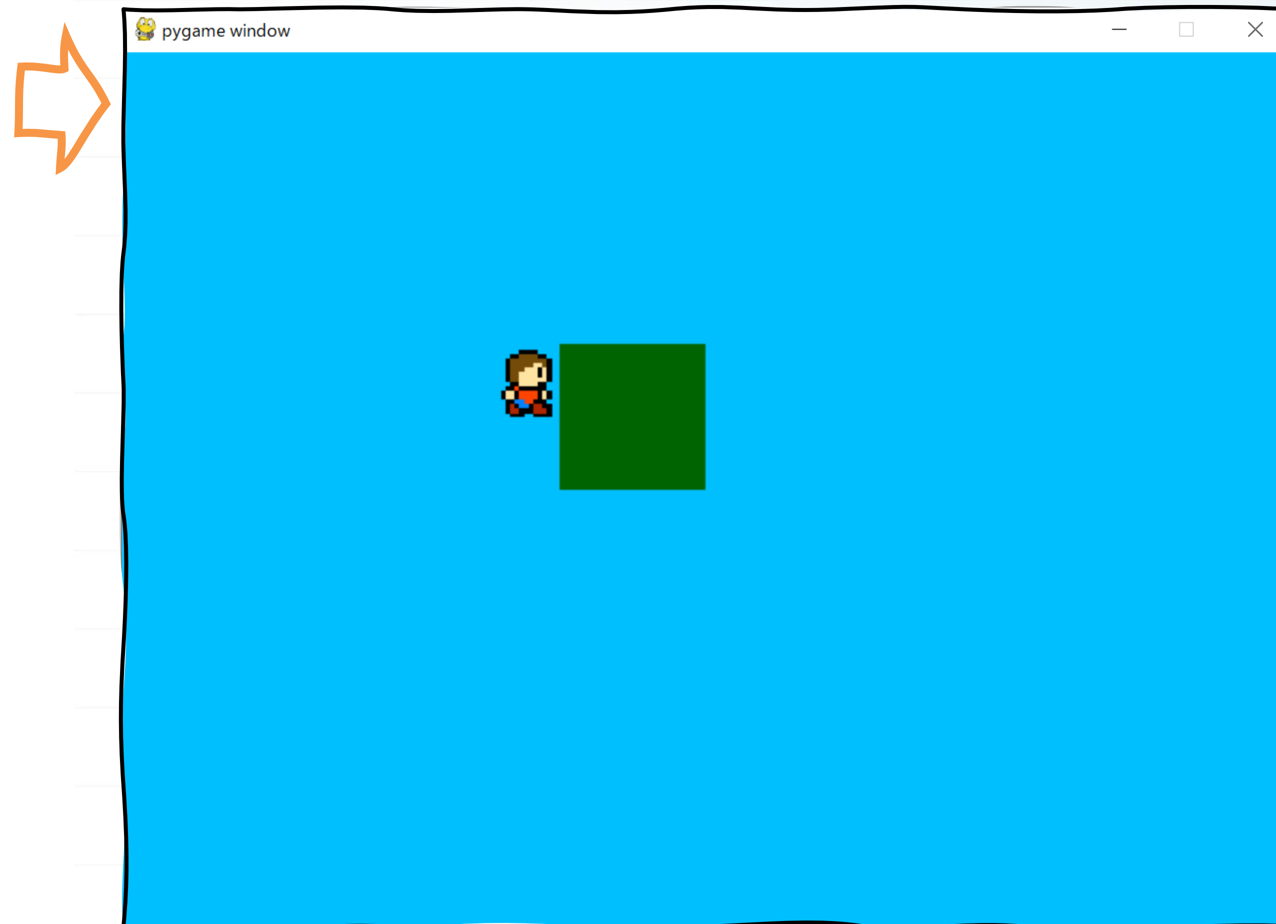
次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

障害物と衝突するプログラムを作る

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

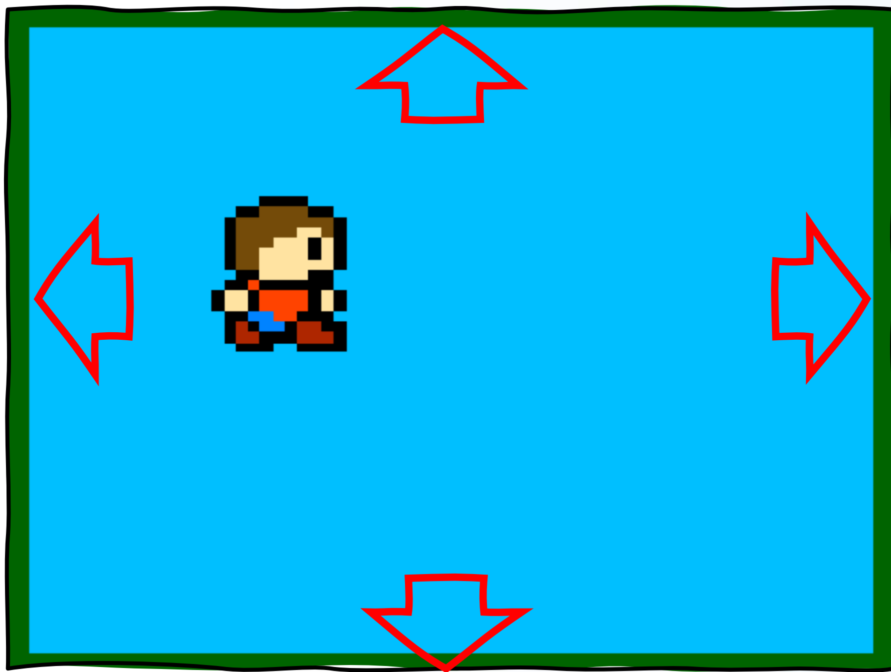
「Run Module」またはF5キーを押してプログラムを実行する。



Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

複数のRectとの衝突判定

次は１つだった障害物を4つに増やし、ステージの上下左右に置いてみよう。これまではすり抜けられていた画面の端にゲームステージの壁を作ることができる。



壁が4つなので「colliderect」関数を4回繰り返してもいいが、こういう時は「collidelist」関数を使うと便利。複数の壁のRectをリストに入れておいて、「このリストのどれかと衝突しているだろうか」と1回で調べることができる。

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

複数のRectとの衝突判定

rectAが、リストの中のどれかのrectと衝突しているか調べる
変数(何番目と衝突したか？) = rectA.collidelist(リスト)

「collidelist」関数は複数のRectをリストに入れて使用する。衝突判定にあたり、「衝突したか、していないか」ではなく、「何番目と衝突したか？」という情報が返ってくる。「どれとも衝突していないとき」は「-1」が返ってくる。逆にいうと、返ってきた値が「-1」でないなら「どれかと衝突している」とわかる。

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ゲームステージの壁を作るプログラム

```
#ゲームの準備をする
import pygame as pg, sys
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)  ・・画像の左右を反転
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),           ・・追加する箇所
         pg.Rect(0,580,800,20)]
#メインループで使う変数
```

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ゲームステージの壁を作るプログラム

```
rightFlag = True
#ゲームステージ
def gamestage():
    global rightFlag
    #画面を初期化する
    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]:
```

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ゲームステージの壁を作るプログラム

```
        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
    if rightFlag:
        screen.blit(myimgR, myrect)
    else:
```

・・追加・修正する箇所

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ゲームステージの壁を作るプログラム

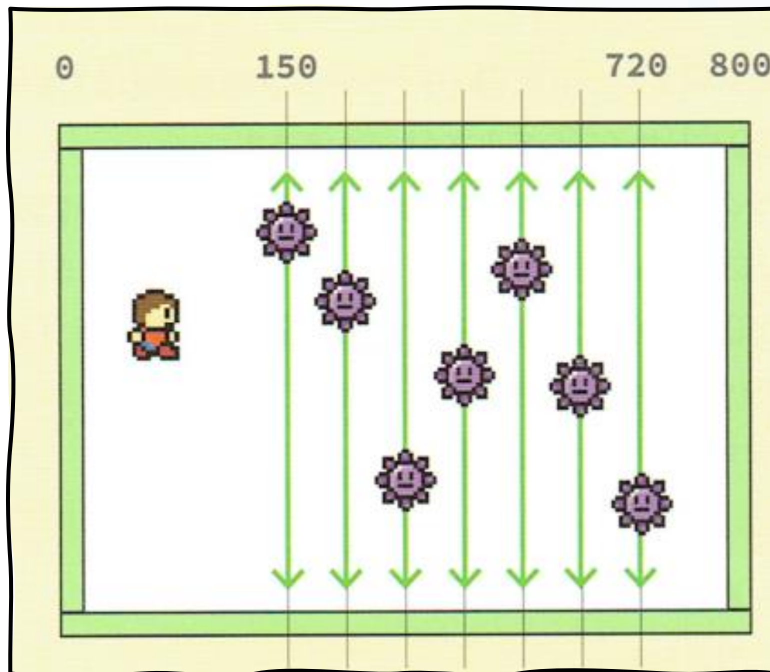
```
        screen.blit(myimgL, myrect)
    #壁の処理
    for wall in walls:
        pg.draw.rect(screen, pg.Color("DARKGREEN"), wall)
#この下をずっとループする
while True:
    gamestage()
    #画面を表示する
    pg.display.update()
    pg.time.Clock().tick(60)
    #閉じるボタンが押されたら、終了する
    for event in pg.event.get():
        if event.type == pg.QUIT:
            pg.quit()
            sys.exit()
```

・・追加・修正する箇所

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナをたくさんばらまく

プレイヤと壁ができたので、今度はワナを作る。完全なランダムでワナをばらまいてしまうと、ワナにとり囲まれて絶対進めない無理ゲーになってしまう。キャラが通り抜けられるよう、適度な隙間を作る。



プレイヤの少し右（150ぐらい）から横方向に一定間隔に配置すると、適度に隙間が空いて進むことができる。その上で、縦方向はランダムにすれば、毎回違うステージを作ることができる。

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナをたくさんばらまく

空っぽのリストを作る
リスト名 = []

リストに値を追加する
リスト名.append(値)

壁データの下にワナデータを追加する。ワナ画像をtrapimgに読み込んで、30 × 30のサイズにする。次にワナの位置データを作るために,trap=[]と命令し、空のリストを作る。for文を使って20個のワナを作る。

【プログラム例】
```
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))
```

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナをばらまくプログラム

```
#ゲームの準備をする
import pygame as pg, sys
import random ・・追加する箇所
pg.init()
screen = pg.display.set_mode((800, 600))・・ゲーム用ウィンドウを作る
#プレイヤーデータ
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

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナをばらまくプログラム

```
#ワナデータ
trapimg = pg.image.load("images/uni.png")
trapimg = pg.transform.scale(trapimg, (30, 30))
traps = [] ・・空っぽのリストを作成
for i in range(20):
    wx = 150 + i * 30
    wy = random.randint(20,550) ・・20から550の間で乱数を作りwyに代入
    traps.append(pg.Rect(wx,wy,30,30)) ・リスト「traps」に要素を追加する
#メインループで使う変数
rightFlag = True
#ゲームステージ
def gamestage():
    global rightFlag
    #画面を初期化する
    screen.fill(pg.Color("DEEPSKYBLUE"))
```

・・追加する箇所

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナをばらまくプログラム

```
vx = 0
vy = 0
#ユーザー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
if key[pg.K_UP]:
    vy = -4
if key[pg.K_DOWN]:
    vy = 4
```

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナをばらまくプログラム

```
#プレイヤーの処理
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
for wall in walls:
    pg.draw.rect(screen, pg.Color("DARKGREEN"), wall)
```

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナをばらまくプログラム

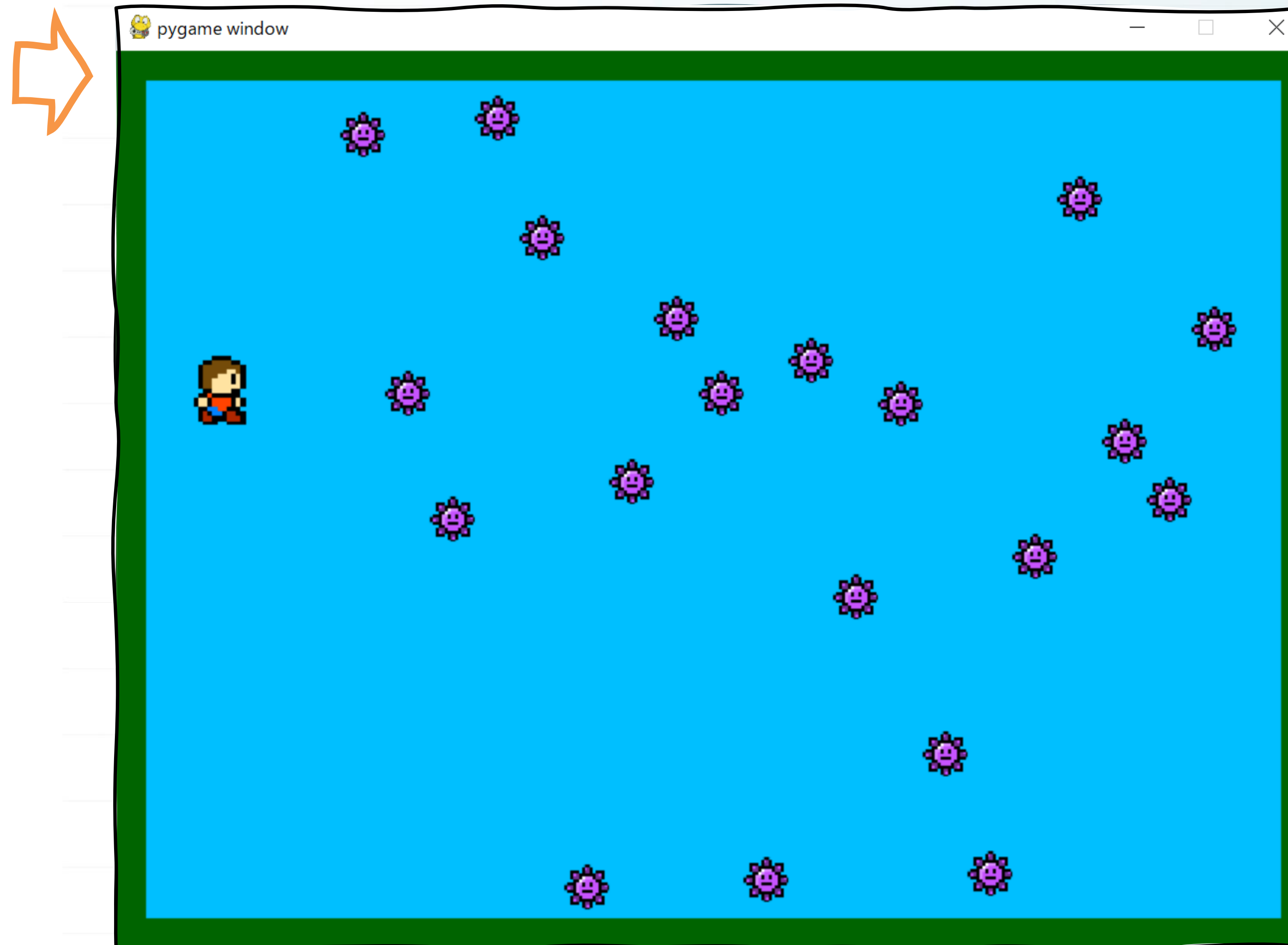
```
    #ワナの処理
    for trap in traps:
        screen.blit(trapimg, trap)
#この下をずっとループする
while True:
    gamestage()
    #画面を表示する
    pg.display.update()
    pg.time.Clock().tick(60)
    #閉じるボタンが押されたら、終了する
    for event in pg.event.get():
        if event.type == pg.QUIT:
            pg.quit()
            sys.exit()
```

・・追加する箇所

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナをばらまくプログラム

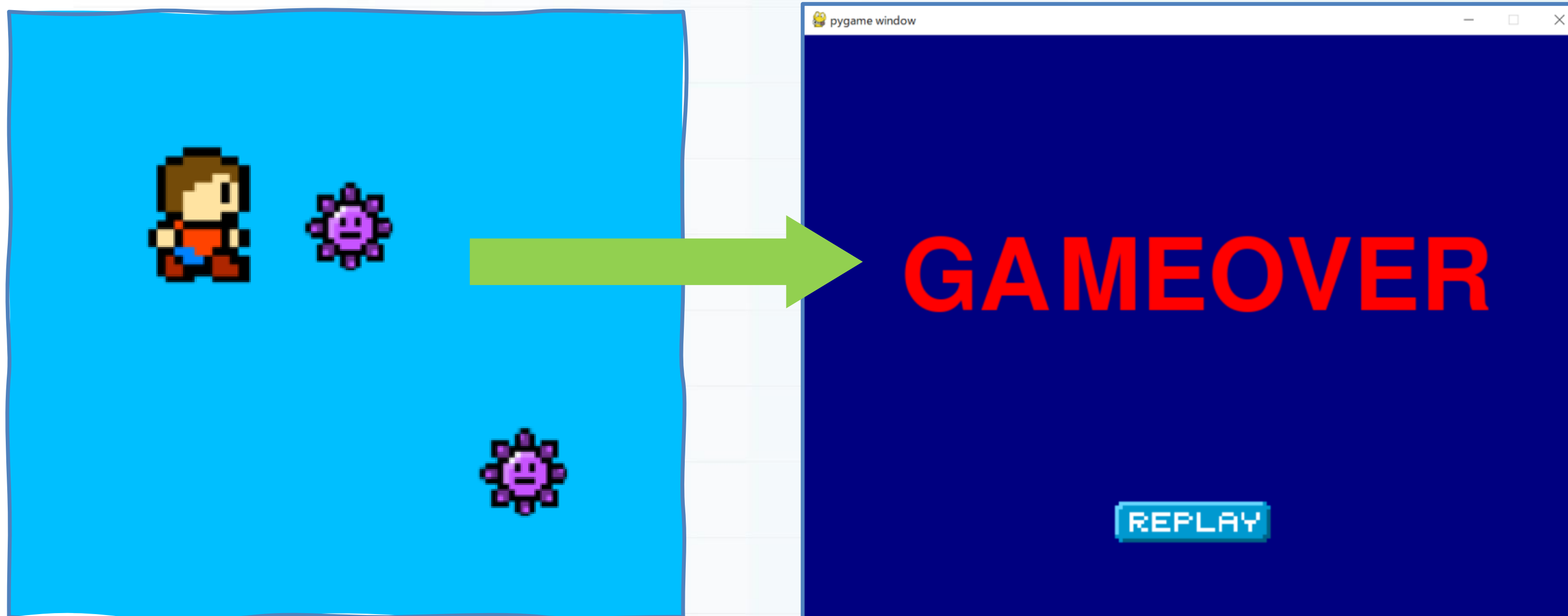
「Run Module」またはF5キーを押してプログラムを実行する。



Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナと衝突したらゲームオーバー

ワナとの衝突判定機能を作る。たくさんの衝突判定を行うのでcollidelist関数を使う。ワナに触れたら一発でゲームオーバーになる仕様を考える。



ワナとの衝突判定は「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」が必要になるので作り、「メインループ」から切り替える。また、何度でもプレイできるように「ゲームリセット関数」も用意する。

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナと衝突したらゲームオーバー

```
#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

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナと衝突したらゲームオーバー

```
#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))  ・・解説1
#ボタンデータ
replay_img = pg.image.load("images/replaybtn.png")
#メインループで使う変数
rightFlag = True
pushFlag = False  ・・ボタンが押されたことをしめすフラグ
page = 1
```

・・追加する箇所（replay_img = pg.image.load("images/replaybtn.png")）

・・追加する箇所（pushFlag = False、page = 1）

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナと衝突したらゲームオーバー

#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() ・・解説2(復習)
    (mx, my) = pg.mouse.get_pos() ・・解説2(復習)
    if mdown[0]: ・・解説2(復習)
        if btn.collidepoint(mx, my) and pushFlag == False: ・・解説3
            pg.mixer.Sound("sounds/pi.wav").play() ・・解説4
            page = newpage
            pushFlag = True
    else:
        pushFlag = False
```

・・追加する箇所

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナと衝突したらゲームオーバー

```
#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]:
        vx = 4
        rightFlag = True
    if key[pg.K_LEFT]:
```

・・追加する箇所（global page）

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナと衝突したらゲームオーバー

```
        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: ・・解説5(復習)
        myrect.x = myrect.x - vx
        myrect.y = myrect.x - vy
    if rightFlag:
        screen.blit(myimgR, myrect)
    else :
```

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナと衝突したらゲームオーバー

```
        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
#データのリセット
def gamereset() :
    myrect.x = 50
    myrect.y = 100
```

・・追加する箇所（if myrect.collidelist(traps) != -1: ～ page = 2）

・・追加する箇所（def gamereset() : ～ myrect.y = 100）

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナと衝突したらゲームオーバー

```
    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)
#この下をずっとループする
while True:
```

・・追加する箇所

・・追加する箇所

・・解説6(復習)

次のページに続く

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナと衝突したらゲームオーバー

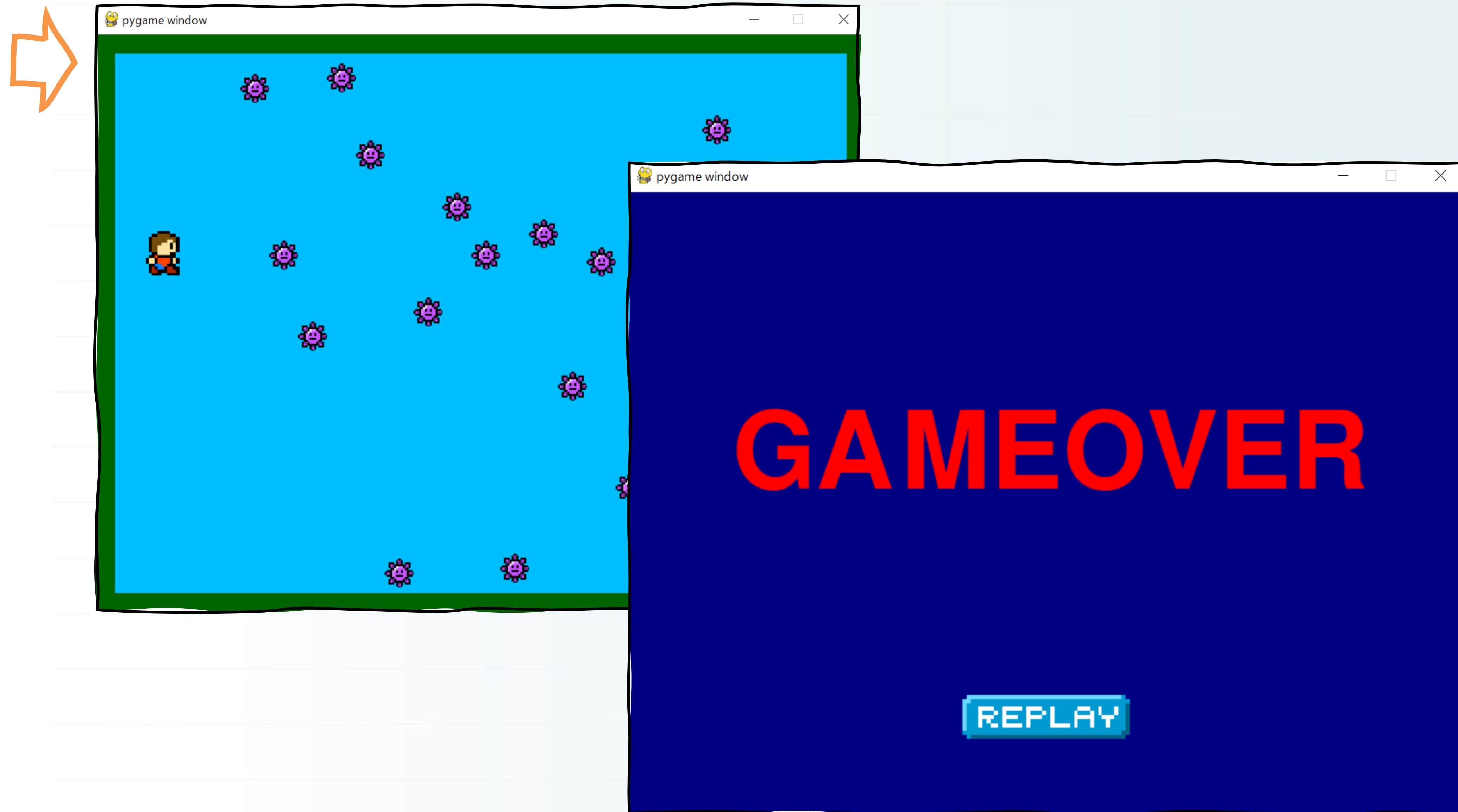
```
if page == 1:
    gamestage()
elif page == 2:
    gameover()
#画面を表示する
pg.display.update()
pg.time.Clock().tick(60)
#閉じるボタンが押されたら、終了する
for event in pg.event.get():
    if event.type == pg.QUIT:
        pg.quit()
        sys.exit()
```

・・追加する箇所

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

ワナと衝突したらゲームオーバー

「Run Module」またはF5キーを押してプログラムを実行する。



Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

解説1:画像をリスト化して使う

同じ画像ファイルを複数の座標に配置したいときにリスト化して使うことができる。

```
画像変数=pygame.image.load(" 画像ファイル.png")
画像変数=pygame.transform.scale(画像変数,(幅,高さ))
```

画像情報

```
リスト名=[]
リスト名.append(pygame.Rect(x座標,y座標,幅,高さ))
```

値

画像情報(座標が異なる)をリストに代入

```
リスト=[
    pygame.Rect(x1,y1,幅,高さ),
    pygame.Rect(x2,y2,幅,高さ),
    pygame.Rect(x3,y3,幅,高さ)
]
```

0
1
2

```
for i in リスト名:
    スクリーン.blit(画像変数, i )
```

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

解説1:画像をリスト化して使う

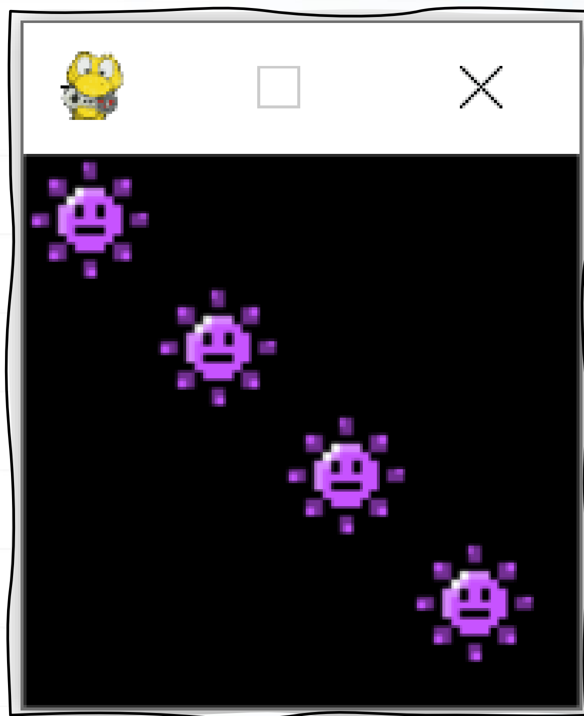
【リストを使用せずに画像を複数表示させる場合】

```
import pygame
import sys
pygame.init()
スクリーン= pygame.display.set_mode((130,130))・ディスプレイサイズ設定
ワナ画像=pygame.image.load("uni.png") ・・画像データのアップロード
ワナ画像=pygame.transform.scale(ワナ画像,(30,30)) ・・画像のサイズ調整
スクリーン.blit(ワナ画像,(0,0)) ・・画像の配置
スクリーン.blit(ワナ画像,(30,30)) ・・画像の配置
スクリーン.blit(ワナ画像,(60,60)) ・・画像の配置
スクリーン.blit(ワナ画像,(90,90)) ・・画像の配置
pygame.display.update()
```

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

解説1:画像をリスト化して使う

「Run Module」またはF5キーを押してプログラムを実行する。



Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

解説1:画像をリスト化して使う

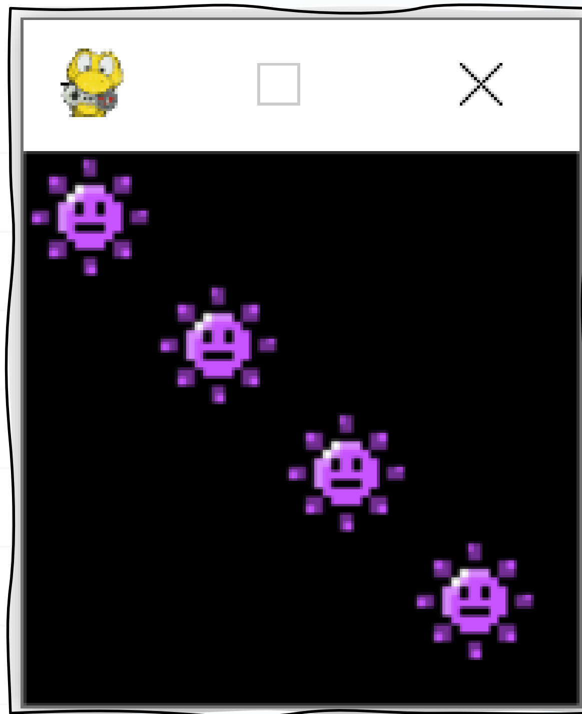
【リストを使用して画像を複数表示させる場合】

```
import pygame
import sys
pygame.init()
スクリーン= pygame.display.set_mode((130,130))・ディスプレイサイズ設定
ワナ画像=pygame.image.load("uni.png") ・・画像データのアップロード
ワナ画像=pygame.transform.scale(ワナ画像,(30,30)) ・・画像のサイズ調整
ワナ=[]・・空のリストを作成
for a in range(4):
    wx = a*30
    wy = a*30
    ワナ.append(pygame.Rect(wx,wy,30,30))・・空のリストに画像情報を追加
for i in ワナ:
    スクリーン.blit(ワナ画像,i) ・・リストの中にある画像情報をすべて表示
pygame.display.update()
```

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

解説1:画像をリスト化して使う

「Run Module」またはF5キーを押してプログラムを実行する。



Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

解説2(復習):マウスで絵を動かす

マウスが押されたかどうかを調べるにはmouse.get_pressed関数を使う。この関数を使うとマウスのボタンが今、押されているかどうかが分かる。さらにmouse.get_pos関数を使うとマウスがどこを指しているかを調べることができる。

マウスが押されたかを調べる
```
マウス変数 = pg.mouse.get_pressed()
(mx,my) = pg.mouse.get_pos()
```

「マウスのどのボタンが押されているか」を調べる「マウス変数 = pg.mouse.get_pressed()」関数では、左ボタンがおされるとマウス変数[0]、右ボタンが押されるとマウス変数[2]が入る。

また、「マウスが画面のどこを指しているか」を調べる「(mx,my) = pg.mouse.get_pos()」関数では、マウスが指している位置がmx,myの２つの変数に入る。

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

解説3(復習):衝突を判定する

衝突判定に使う関数について再確認する。

ある点(x,y)がbtnの範囲内にあるかを調べる
変数(範囲内にあるかないか？) = btn.collidepoint(x,y)

```
def button_to_jump(btn, newpage):
    mdown = pg.mouse.get_pressed()
    (mx, my) = pg.mouse.get_pos()
    if mdown[0]:
        if btn.collidepoint(mx, my):
            page = newpage
```

・・・・button_to_jump()関数の引数(btn)がマウスポインタに触れたかを調べている。

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

解説3(復習):衝突を判定する

【本文を一部抜粋】

```
def button_to_jump(btn, newpage):
    mdown = pg.mouse.get_pressed()
    (mx, my) = pg.mouse.get_pos()
    if mdown[0]:
        if btn.collidepoint(mx, my):
            page = newpage


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))

    button_to_jump(btn1, 2)
```

・・button_to_jump()関数にbtn=btn1かつpage=2をいれた状態で実行している

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

解説4:効果音を鳴らす

効果音ファイルを鳴らすにはmixer.Sound関数を使う。画像ファイルを読み込むときと同じようにサウンドファイルを読み込ませて使用する。

効果音を鳴らす
pg.mixer.Sound(“ サウンドファイルパス”).play()

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

解説5(復習):複数のRectとの衝突判定

rectAが、リストの中のどれかのrectと衝突しているか調べる
変数(何番目と衝突したか？) = rectA.collidelist(リスト)

「collidelist」関数は複数のRectをリストに入れて使用する。衝突判定にあたり、「衝突したか、していないか」ではなく、「何番目と衝突したか？」という情報が返ってくる。「どれとも衝突していないとき」は「-1」が返ってくる。逆にいうと、返ってきた値が「-1」でないなら「どれかと衝突している」とわかる。

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

解説6(復習):図形や文字、画像を描く

pygameで文字列を表示させるには、文字を画像にしてからその画像を描画させる必要がある。

文字列を表示手順
①フォントを用意する
②そのフォントを使って、文字列の画像を作る
③その画像を描画する

フォントを準備する

```
font = pg.font.Font(None, 文字サイズ)
```

※デフォルトのフォントを指定する場合、Noneを使う

文字列の画像を作る

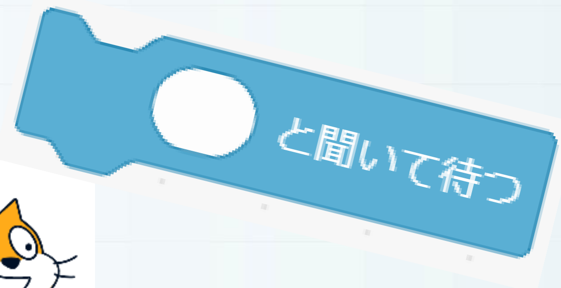
```
画像変数 = font.render(文字列, True, 色)
```

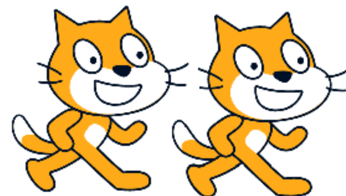
※render 与える
※文字の境界を滑らかにするか
⇒True（滑らかにする）、False（かくかく）

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

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。





Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

メモ

Copyright 2020 テクノロ Co.,Ltd All Rights Reserved

プログラミング教室の
テクノロ

なまえ：

Copyright 2020 テクノロ Co.,LtdAll Rights Reserved