プログラミング教室のテクノロ

Pythonの道

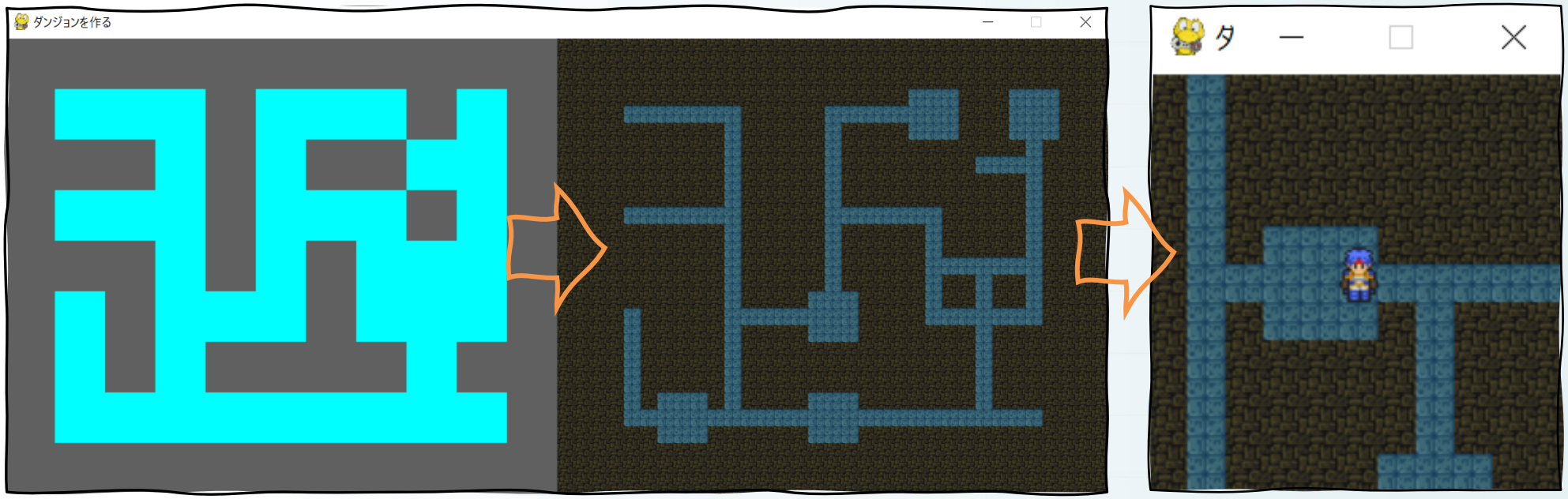
本格RPGを作る(前編)

もくじ

・迷路を自動生成する

・迷路をダンジョンにかえる

・ダンジョン内を移動する

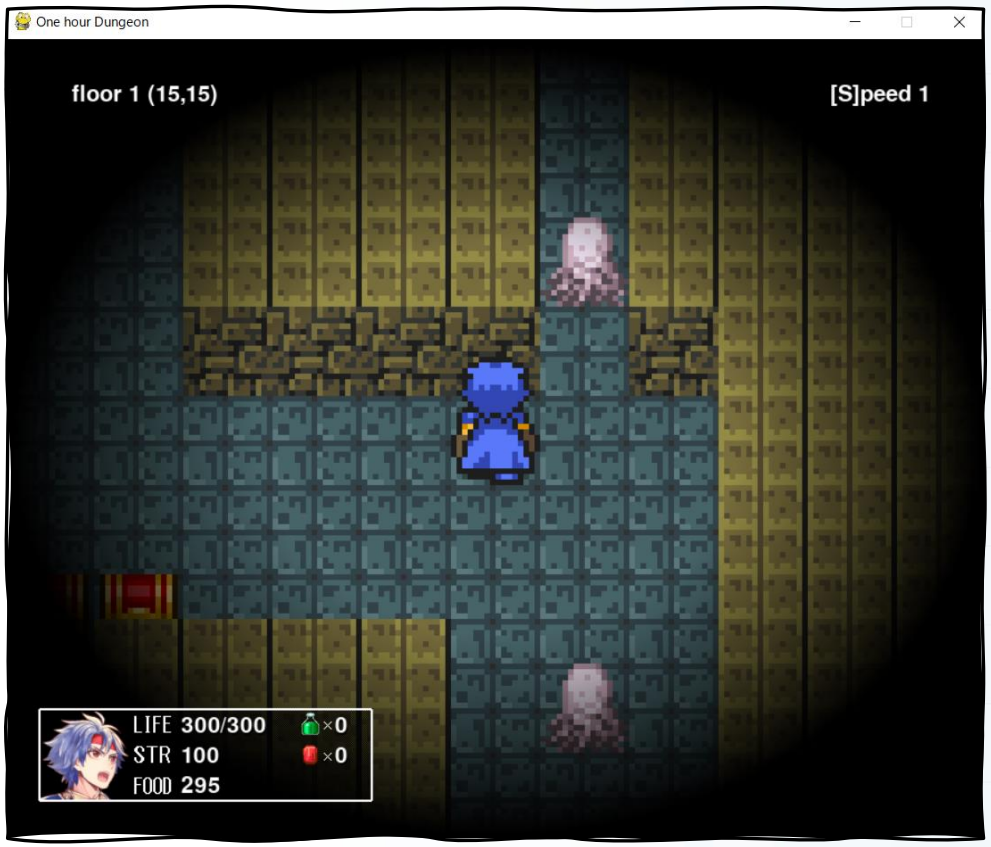


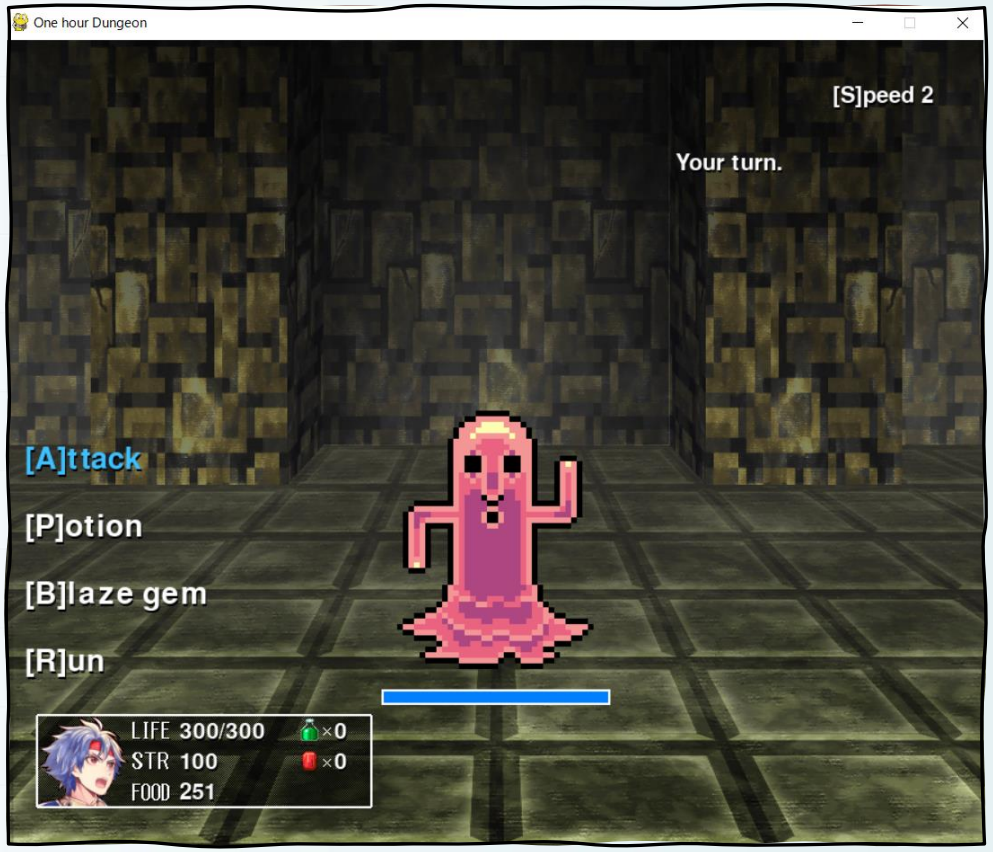
ロールプレイングゲームについて

ロールプレイングゲームとは主人公やその仲間達を成長させながら冒険するタイプのゲームをいう。
元々はコンピュータゲームではなく、数人でテーブルを囲み、サイコロや紙と鉛筆を使って一定のルールを設けて遊ぶテーブルゲームを差していた。
ロールプレイングゲームの代表作には「ドラゴンクエスト」や「ファイナルファンタジー」がある。

制作するRPGの特徴

RPGはゲーム制作の中でも難易度が高いと言われている。
今回作成するRPGには以下の特徴を盛り込む。
★自動生成されるダンジョンを探索し、到達できた最大階層数を競う。
★戦闘シーンはコマンドを入力して敵と戦う画面構成にする。





今回作成するゲームのルール

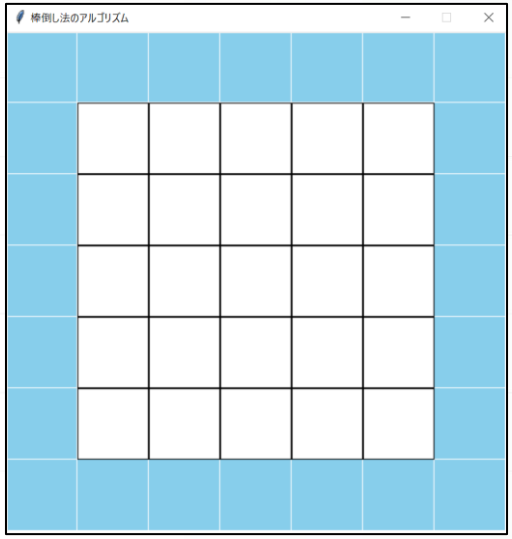
今回作成するゲームのルールについて解説する。
①移動シーン
　☆自動生成されるダンジョン内を方向キーで移動する
　　・移動すると食料が減り、食料がある間は、歩くごとにライフが回復する
　　・食料が0になると、歩くごとにライフが減り、ライフが0になるとゲームオーバー
　　・ダンジョンには宝箱とまゆがあり、戦闘中に使えるアイテムやモンスターが入っている
　　・下り階段から次の階層に移動し、到達できた階層数を競う
②戦闘シーン
　☆プレイヤの行動と敵の行動が交互に行われるターン制とする
　　・プレイヤはコマンドを選び戦闘を行う
　　・敵の攻撃を受け、ライフが0になるとゲームオーバ

迷路を自動生成する

まず最初に移動シーンのダンジョン作成から始める。今回はダンジョンそのものを自動生成するアルゴリズムを考える。
コンピュータにランダムでダンジョンを作成するにはどうすればよいか？ダンジョンを作るアルゴリズムは昔から色々なものが考え出されている。その中で有名な棒倒し法と呼ばれるアルゴリズムを使ってみる。

棒倒し法のアルゴリズム

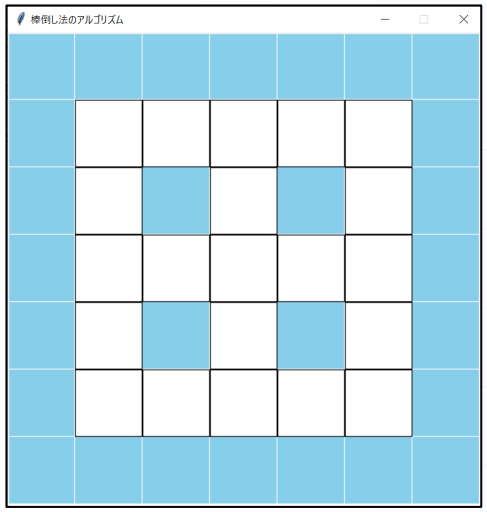
①周りを壁で埋める。青いマスが壁、白いマスが床とする。内部はすべて床とする。



「棒倒し法」以外にも「壁伸ばし法」「穴掘り法」があることを知っておこう。
「棒倒し法」が一番簡単だが、シンプルな迷路しかつくれない。

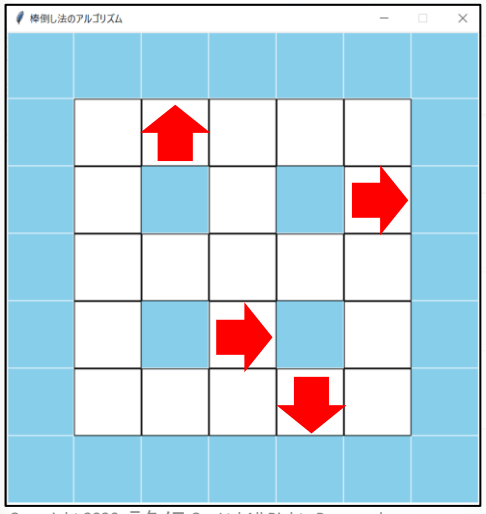
迷路を自動生成する

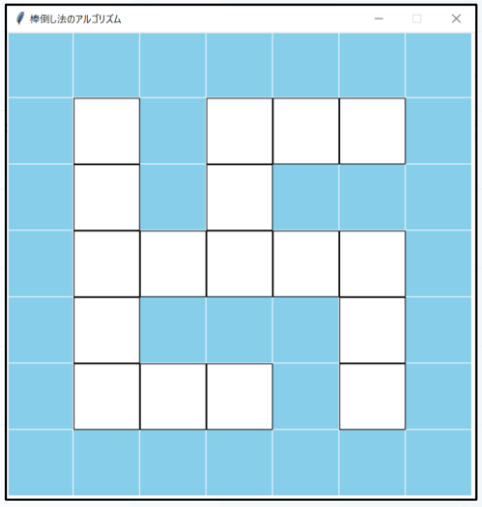
②次に内部に1マスおきに柱を設ける。
※柱と表現しているが、壁と同じ意味



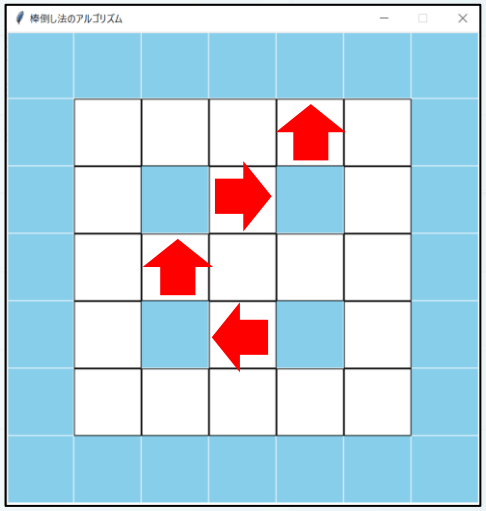
③それぞれの柱から上下左右いずれかの方向（ランダム）に壁を作る。

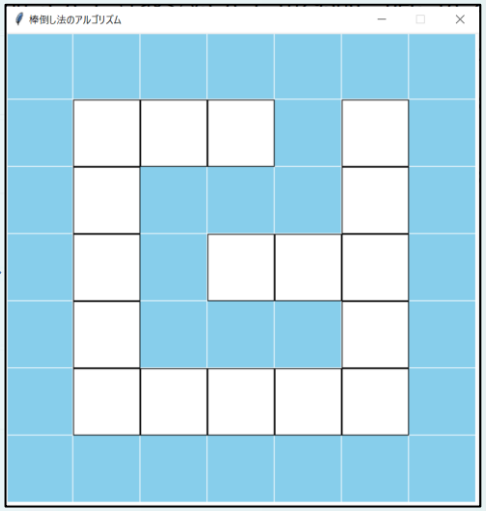
【例1】





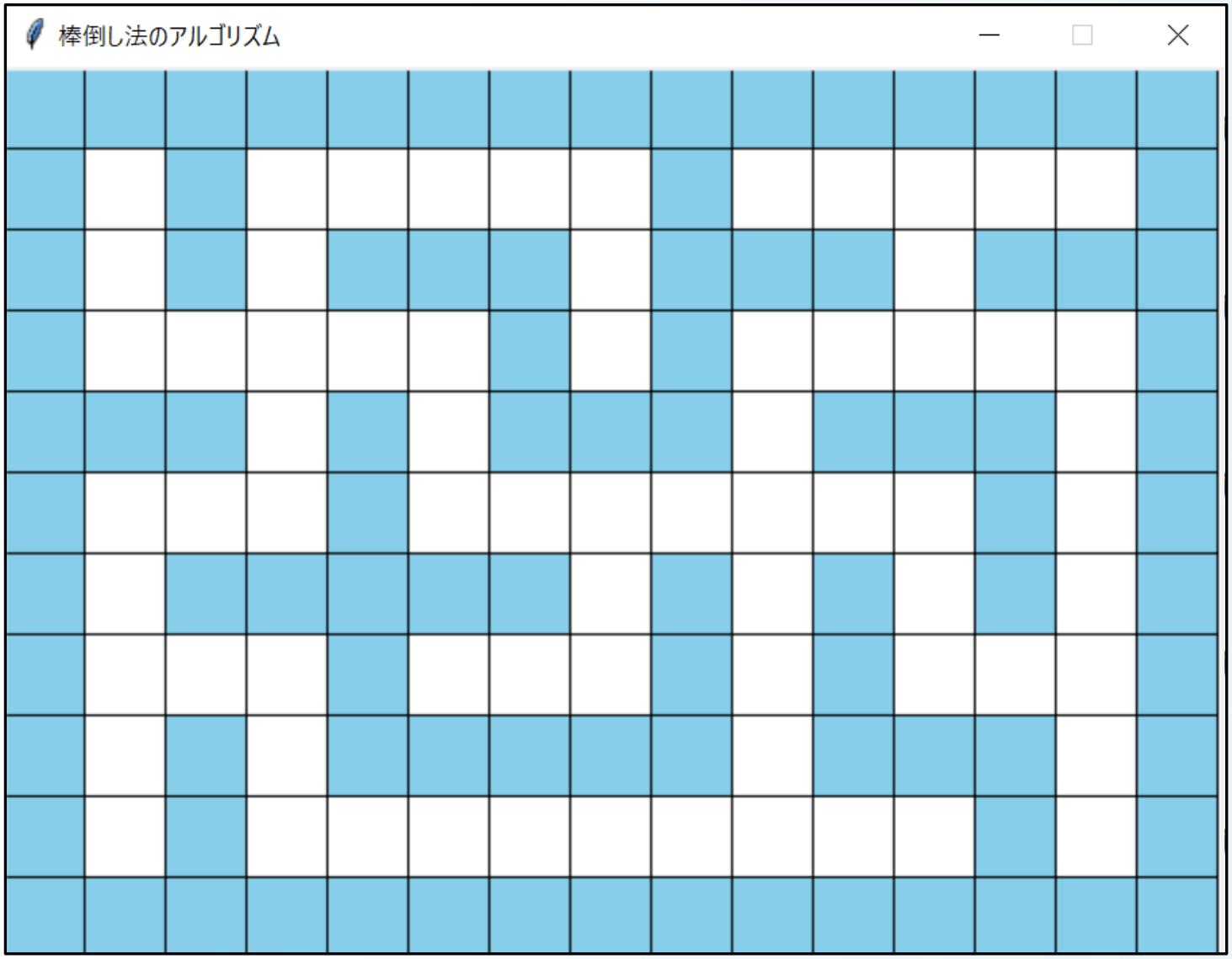
【例2】





迷路を自動生成する

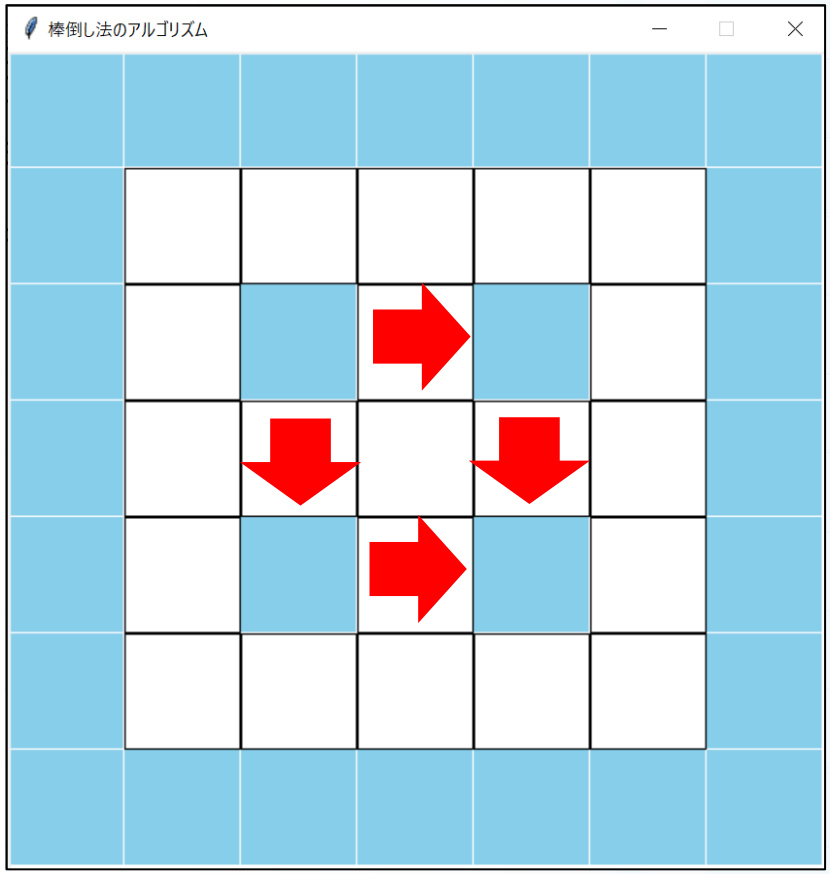
④このアルゴリズムを使って横15マス、縦11マスの迷路を作ると以下のようになる。



迷路を自動生成する

⑤棒倒し法の注意点

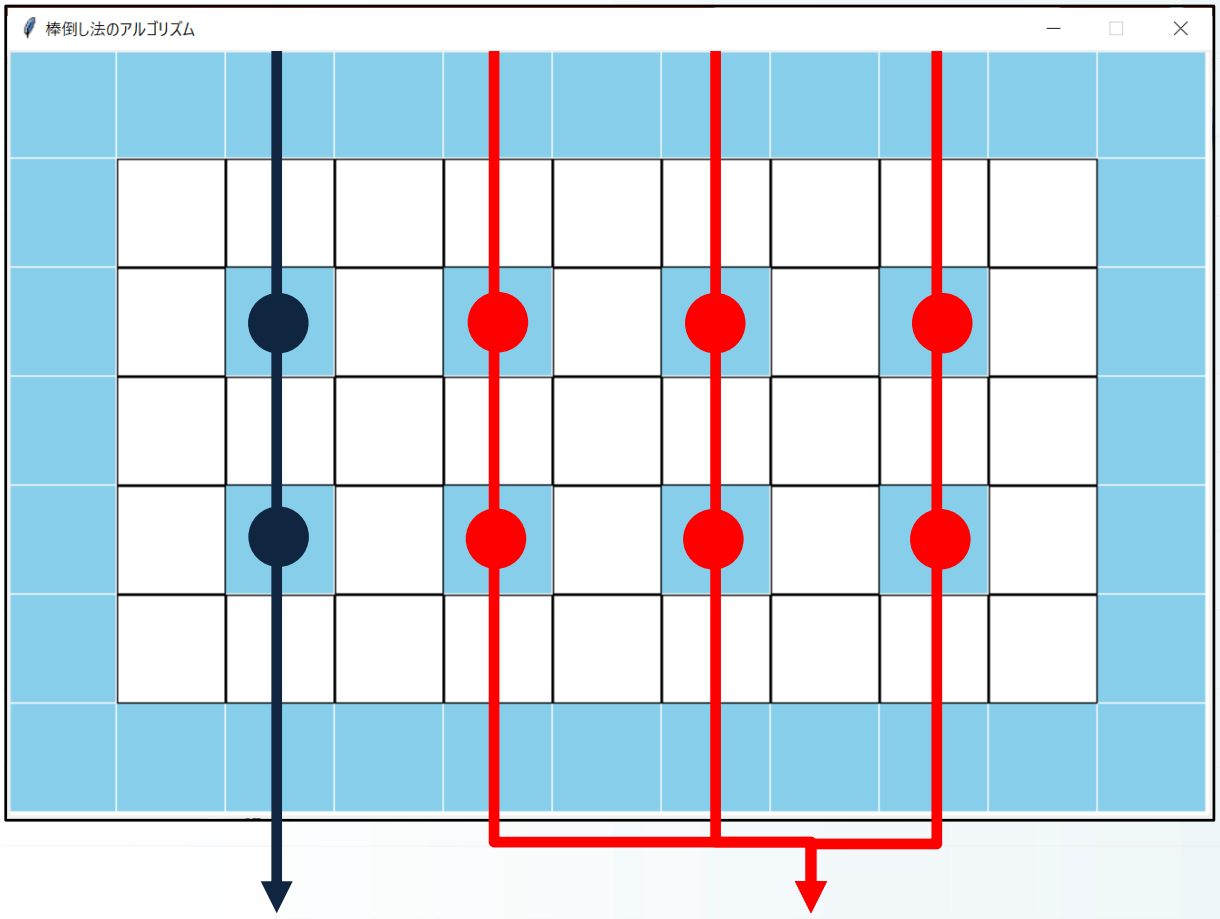
ランダムに4方向に壁を作ると、入れない場所ができる恐れがある。例えば下のように壁が作られた場合、中央部分に入れなくなる。



迷路を自動生成する

⑥棒倒し法の解決法

一番左の列の柱からは4方向いずれかに壁を作り、その次の列からは、上、下、右の3方向いずれかに壁を作る。



迷路を自動生成する

棒倒し法で迷路を作るプログラムを作る。

```
import pygame
import sys
import random
#色の作成(RGBカラーで設定)
CYAN = (0,255,255)
WHITE = (255,255,255)
#空っぽのリストを作成
maze = []
#空っぽのリストに値がすべて0のリストを追加して、二次元リストを生成
for y in range(9):
    maze.append([0,0,0,0,0,0,0,0,0,0,0])
#周りの壁
def make_maze():
```

次のページに続く

迷路を自動生成する

```
XP = [0,1,0,-1]  #上下左右に柱をセットする二次元リスト
YP = [-1,0,1,0]    XPとYPのセットの座標位置に柱がセットされる
#左右の壁を作成
for x in range(11):
    maze[0][x]= 1
    maze[8][x]= 1
#上下の壁を作成
for y in range(1,8):
    maze[y][0]= 1
    maze[y][10]=1
#中を何もない状態に
for y in range(1,8):
    for x in range(1,10):
        maze[y][x] = 0
#柱
```

次のページに続く

迷路を自動生成する

```
    for y in range(2,7,2):
        for x in range(2,7,2):
            maze[y][x]=1
```

解説1

```
    #柱から上下左右に壁を作る
    for y in range(2,7,2):
        for x in range(2,9,2):
            d = random.randint(0,3)
            #二列目からは左に壁を作らない。xが3以上のときにこの処理に入る。
            if x > 2:
                d = random.randint(0,2)
            maze[y+YP[d]][x + XP[d]] = 1
def main():
    pygame.init()
    pygame.display.set_caption("迷路を作る")
    screen = pygame.display.set_mode((528,432))
```

次のページに続く

迷路を自動生成する

```
make_maze()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE:
                make_maze()
    for y in range(9):
        for x in range(11):
            X = x*48
            Y = y*48
            if maze[y][x]==0:
                pygame.draw.rect(screen,WHITE,[X,Y,48,48])
```

次のページに続く

迷路を自動生成する

```
                    if maze[y][x]==1:
                        pygame.draw.rect(screen,CYAN,[X,Y,48,48])
            pygame.display.update()
main()
```

解説1(乱数で壁をつくる)

棒倒し法と乱数を使って壁を作るアルゴリズムを確認する。

```
XP = [0 , 1 , 0 , -1]
YP = [-1, 0 , 1 , 0 ]
```

上に壁を作る　右に壁を作る　下に壁を作る　右に壁を作る　.　.　.

```
for y in range(2,7,2):
    for x in range(2,9,2):
        d = random.randint(0,3)
        if x > 2:
            d = random.randint(0,2)
        maze[y+YP[d]][x + XP[d]] = 1
```

解説1(乱数で壁をつくる)

maze[y+YP[d]][x + XP[d]] = 1の動きを書き出してみよう。

【例】※具体的な数字をいれてみて考える

YP = [-1,0,1,0]

XP = [0,1,0,-1]

```
for y in range(2,7,2):
    for x in range(2,9,2):
```

for文の1回目の処理(すべてのパターンを記載)

d=0の時　maze[2+YP[0]][2+XP[0]] ⇒ maze[2-1][2+0] ⇒ maze[1][2]=1

d=1の時　maze[2+YP[1]][2+XP[1]] ⇒ maze[2+0][2+1] ⇒ maze[2][3]=1

d=2の時　maze[2+YP[2]][2+XP[2]] ⇒ maze[2+1][2+0] ⇒ maze[3][2]=1

d=3の時　maze[2+YP[3]][2+XP[3]] ⇒ maze[2+0][2-1] ⇒ maze[2][1]=1

```
[1,1,1,1,1,1,1,1,1,1,1]
[1,0,1,0,0,0,0,0,0,0,1]
[1,1,1,1,1,0,1,0,1,0,1]
[1,0,1,0,0,0,0,0,0,0,1]
[1,0,1,0,1,0,1,0,1,0,1]
[1,0,0,0,0,0,0,0,0,0,1]
```

dの値はランダムで決まるため、4つの内
どこか1か所だけ１になる

解説1(乱数で壁をつくる)

maze[y+YP[d]][x + XP[d]]の動きを書き出してみよう。

for文の2回目の処理(すべてのパターンを記載)

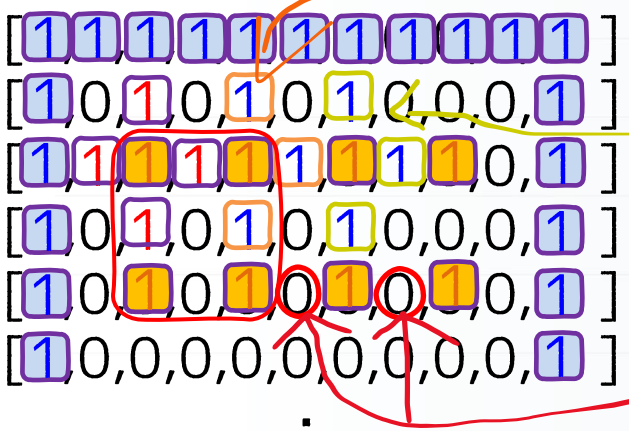
YP = [-1,0,1,0]
XP = [0,1,0,-1]

```
for y in range(2,7,2):
    for x in range(2,9,2):
```

d=0の時　maze[2+YP[0]][4+XP[0]] ⇒ maze[2-1][4+0] ⇒ maze[1][4]=1
d=1の時　maze[2+YP[1]][4+XP[1]] ⇒ maze[2+0][4+1] ⇒ maze[2][5]=1
d=2の時　maze[2+YP[2]][4+XP[2]] ⇒ maze[2+1][4+0] ⇒ maze[3][4]=1

for文の3回目の処理(すべてのパターンを記載)

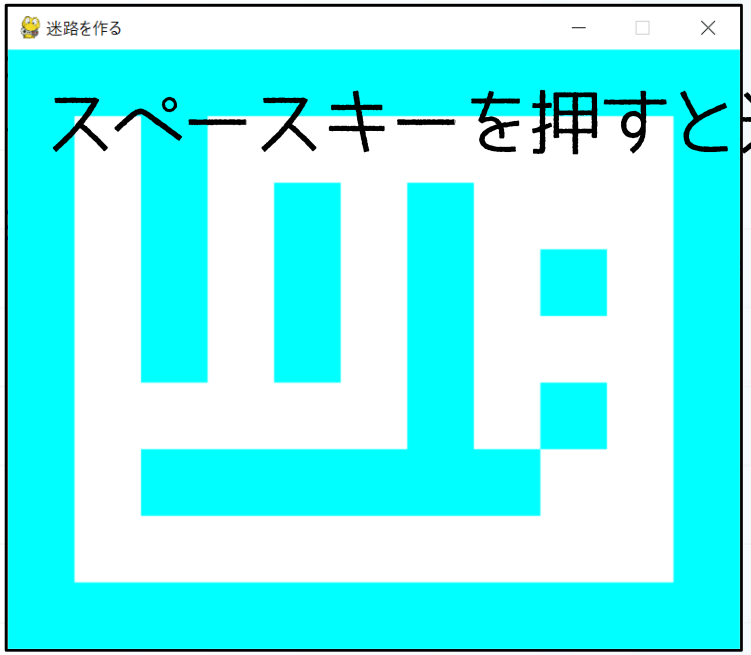
d=0の時　maze[2+YP[0]][4+XP[0]] ⇒ maze[2-1][6+0] ⇒ maze[1][6]=1
d=1の時　maze[2+YP[1]][4+XP[1]] ⇒ maze[2+0][6+1] ⇒ maze[2][7]=1
d=2の時　maze[2+YP[2]][4+XP[2]] ⇒ maze[2+1][6+0] ⇒ maze[3][6]=1



こちら側に壁を作ることが許されていない
ので入れない空間が生まれることはない。

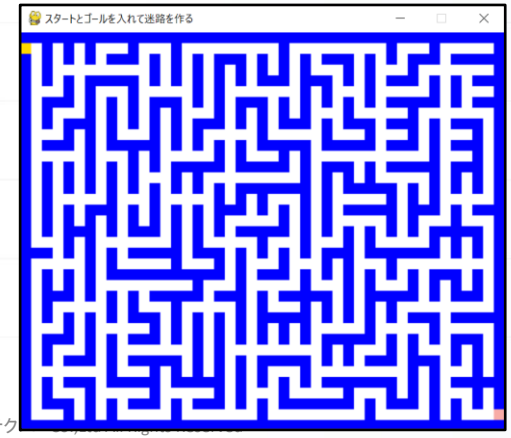
迷路を自動生成する

「Run Module」またはF5キーを押してプログラムを実行する。



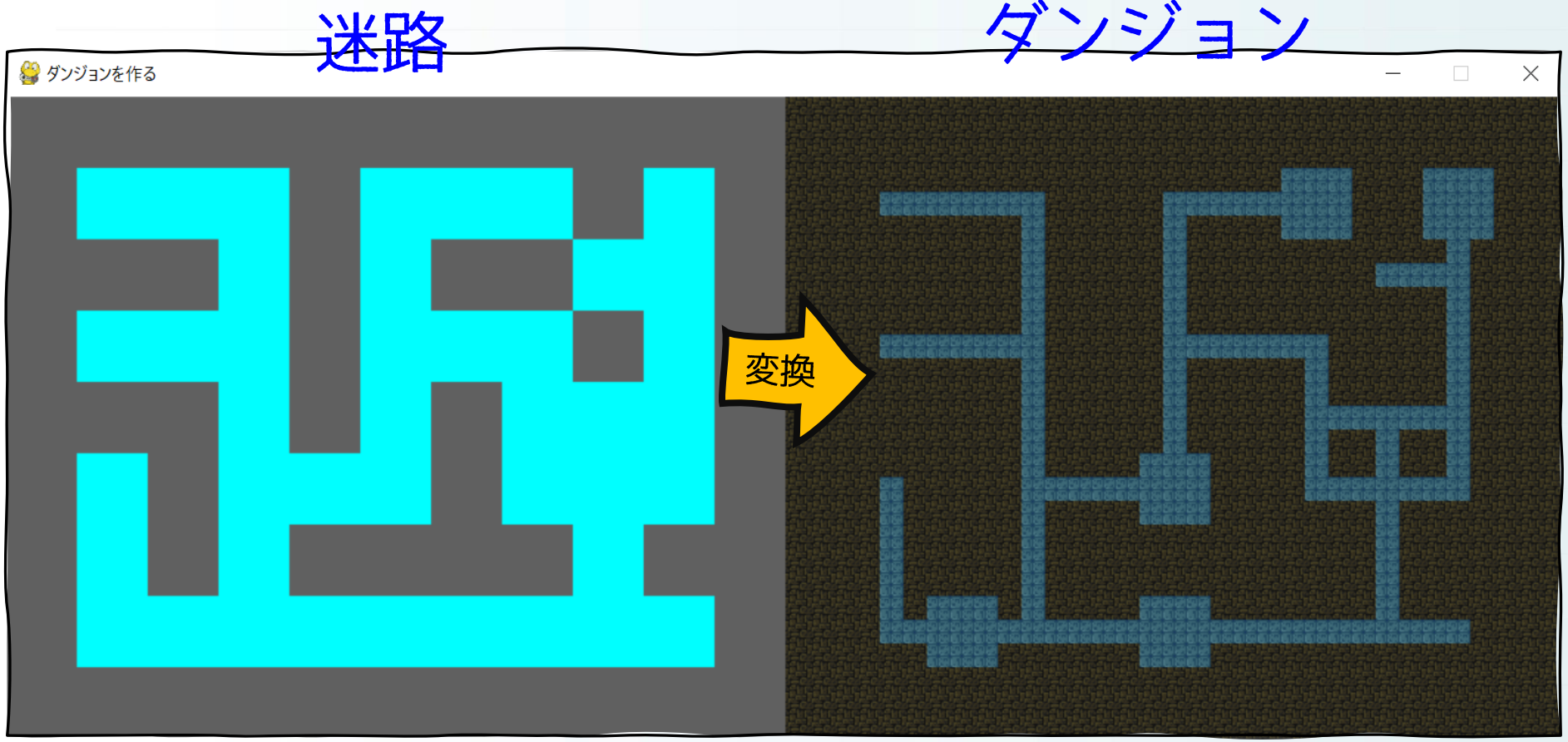
スペースキーを押すと迷路が再生成される。

完成したら理解度確認テストにチャレンジしよう!



迷路をダンジョンに変える。

迷路が完成したら、迷路データをもとにダンジョンを作成する。このテキストでは鉛筆で書いたようなシンプルな通路を迷路と呼び、ゲーム内で探索する地下迷宮をダンジョンと呼ぶこととする。迷路からダンジョンへのデータ変換を行うアルゴリズムを考える。



迷路をダンジョンに変える。

迷路からダンジョンへのデータ変換は次の手順で行う。

■変換手順

・ダンジョンを定義するための二次元リスト「dungeon」を用意する

・mazeのマスの状態を調べながら、「dungeon」に値をセットする

・「dungeon」の中身（要素）を、いったん全て壁にする

・maze[y][x]の値を調べ、0（床）であればランダムに「dungeon」に部屋を作る

・部屋を作らない場合、maze[y][x]の上下左右のマスを調べ、0ならその方向に「dungeon」に通路を作る

迷路をダンジョンに変える。

迷路をダンジョンに変えるプログラムを作る。

```
import pygame
import sys
import random
#色の作成(RGBカラーで設定)
CYAN = (0,255,255)
WHITE = (255,255,255)
#空っぽのリストを作成(迷路)
maze = []
for y in range(9):
    maze.append([0,0,0,0,0,0,0,0,0,0,0])
#空っぽのリストを作成(ダンジョン)
dungeon = []
```

次のページに続く

迷路をダンジョンに変える。

```
for y in range(27):
    dungeon.append([0]*33)  #[0,0,0……,0]と同じ意味。
kabe = pygame.image.load(r"image¥wall.png")
yuka = pygame.image.load(r"image¥floor.png")
#ダンジョンを作る関数
def make_dungeon():#最初に迷路を作成する
    XP = [0,1,0,-1]
    YP = [-1,0,1,0]
    #周りの壁(一番上と一番下の壁)
    for x in range(11):
        maze[0][x]= 1
        maze[8][x]= 1
    #周りの壁(一番左と一番右の壁)
```

33個

次のページに続く

迷路をダンジョンに変える。

```
for y in range(1,8):
    maze[y][0]= 1
    maze[y][10]=1
#中を何もない状態に
for y in range(1,8):
    for x in range(1,10):
        maze[y][x] = 0
#柱を作る
for y in range(2,7,2):
    for x in range(2,7,2):
        maze[y][x]=1
#柱から上下左右に壁を作る
for y in range(2,7,2):
    for x in range(2,9,2):
        d = random.randint(0,3)
```

次のページに続く

迷路をダンジョンに変える。

```
            if x > 2:
                d = random.randint(0,2)
                maze[y+YP[d]][x + XP[d]] = 1
    #上記で作成した迷路からダンジョンを作る
    #ダンジョン全体を壁にする
    for y in range(27):
        for x in range(33):
            dungeon[y][x]=9
    #部屋と通路の配置
    for y in range(1,8):    ・・・解説1
        for x in range(1,10):   #dx,dy:ダンジョンのx座標、y座標
            dx = x*3+1   #迷路の1マスがダンジョンでは3 × 3マスになる。＋１して
            dy = y*3+1    いるのはdx,dyを3 × 3マスの中央の座標にするため
#部屋を作る if maze[y][x] == 0:    #0~99までの数字をランダムに出力すると20%の確
                                    立で20より小さくなる。20%という条件を作る。
                if random.randint(0,99)<20:
```

次のページに続く

迷路をダンジョンに変える。

```
                for ry in range(-1,2): #ry:roomのy座標
                    for rx in range(-1,2): #rx:roomのx座標
                        dungeon[dy+ry][dx+rx]=0 #0:通路
            else: #通路を作る
                dungeon[dy][dx]= 0
                if maze[y-1][x]==0:
                    dungeon[dy-1][dx] = 0
                if maze[y+1][x] == 0:
                    dungeon[dy+1][dx]=0
                if maze[y][x-1] == 0:
                    dungeon[dy][dx-1] = 0
                if maze[y][x+1] == 0:
                    dungeon[dy][dx+1] = 0
def main():
    pygame.init()
```

次のページに続く

迷路をダンジョンに変える。

```
pygame.display.set_caption("ダンジョンを作る")
screen = pygame.display.set_mode((1056,432))
make_dungeon()
#ループ処理
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE:
                make_dungeon()
    #確認用の迷路を表示
    for y in range(9):
        for x in range(11):
```

次のページに続く

迷路をダンジョンに変える。

```
                X = x*48
                Y = y*48
                if maze[y][x]==0:
                    pygame.draw.rect(screen,WHITE,[X,Y,48,48])
                if maze[y][x]==1:
                    pygame.draw.rect(screen,CYAN,[X,Y,48,48])
    #ダンジョンを描画する
    for y in range(27):
        for x in range(33):
            X = x*16+528
            Y = y*16
            if dungeon[y][x] == 0:
                screen.blit(yuka,[X,Y])
            if dungeon[y][x] == 9:
                screen.blit(kabe,[X,Y])
```

次のページに続く

迷路をダンジョンに変える。

```
            pygame.display.update()

main()
```

解説1

迷路をもとにダンジョンを描画するアルゴリズム

迷路の二次元リスト

```
[[1,1,1,1,1,1,1,1,1,1,1],
 [1,0,0,0,1,0,0,0,0,0,1],
 [1,1,1,0,1,0,1,0,0,1,1],
 [1,0,0,0,0,0,1,0,0,0,1],
 [1,0,1,0,1,1,1,1,0,1,1],
 [1,0,1,0,1,0,0,0,0,0,1],
 [1,0,1,0,1,0,1,0,0,1,1],
 [1,0,0,0,0,0,1,0,0,0,1],
 [1,1,1,1,1,1,1,1,1,1,1]]
```

ダンジョンの二次元リスト

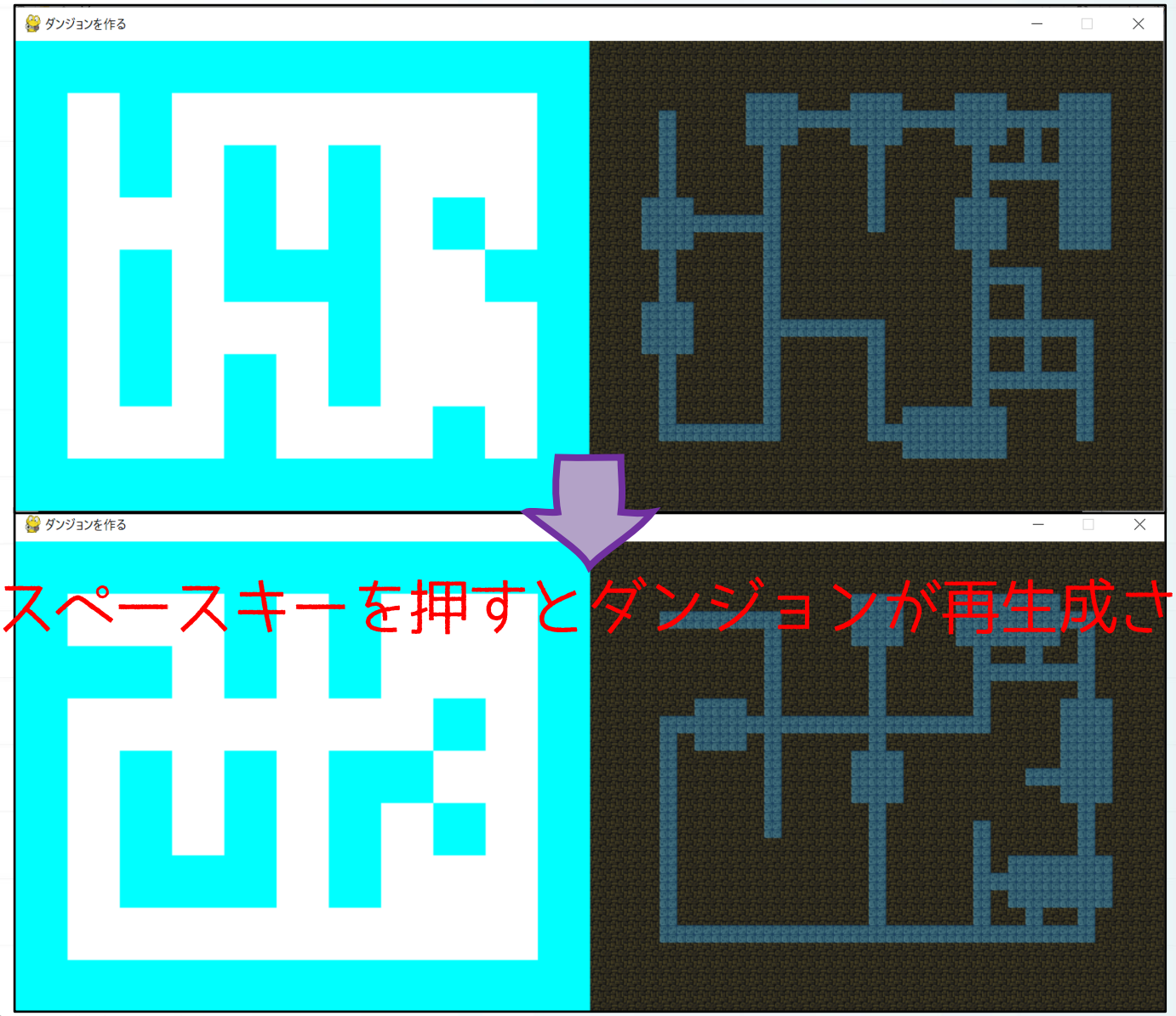
```
[[9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 9, 9, 9, 9, 9, 0, 0, 0, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 9, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 9, 9],
 [9, 9, 9, 9, 9, 9, 9, 9, 9, 0, 0, 0, 9, 9, 9, 9, 0, 9, 9, 9, 9, 9, 0, 9, 9, 0, 9, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 0, 9, 9, 9, 9, 9, 0, 9, 9, 9, 9, 9, 0, 9, 0, 0, 0, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 0, 9, 9, 9, 9, 9, 0, 9, 9, 9, 9, 9, 0, 0, 0, 0, 0, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 0, 9, 9, 9, 9, 9, 0, 9, 9, 9, 9, 9, 0, 9, 0, 0, 0, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 9, 9, 9, 9, 9, 0, 0, 0, 9, 9, 9, 0, 0, 0, 9, 9, 9, 0, 0, 0, 9, 0, 9, 0, 0, 0, 9, 9, 9],
 [9, 9, 9, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 9],
 [9, 9, 9, 9, 0, 9, 9, 9, 9, 0, 0, 0, 9, 9, 9, 0, 0, 0, 9, 9, 9, 0, 0, 0, 9, 0, 9, 0, 0, 0, 9, 9, 9],
 [9, 9, 9, 9, 0, 9, 9, 9, 9, 0, 0, 0, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 0, 9, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 0, 9, 9, 9, 9, 0, 0, 0, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 0, 9, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 0, 9, 9, 9, 9, 0, 0, 0, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 0, 9, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 0, 0, 0, 9, 9, 9, 9, 0, 9, 9, 9, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 9, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 0, 0, 0, 9, 9, 9, 9, 0, 9, 9, 9, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 9, 9],
 [9, 9, 9, 0, 0, 0, 9, 9, 9, 9, 0, 9, 9, 9, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 9, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 0, 9, 9, 9, 9, 9, 0, 9, 9, 9, 9, 0, 0, 0, 9, 9, 9, 9, 0, 9, 0, 0, 0, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 0, 9, 9, 9, 9, 9, 0, 9, 9, 9, 9, 0, 0, 0, 9, 9, 9, 9, 0, 0, 0, 0, 0, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 0, 9, 9, 9, 9, 9, 0, 9, 9, 9, 9, 0, 0, 0, 9, 9, 9, 9, 0, 9, 0, 0, 0, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 0, 9, 9, 9, 9, 9, 0, 9, 0, 0, 0, 9, 0, 9, 9, 9, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 9],
 [9, 9, 9, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 9, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 9],
 [9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 0, 0, 0, 9, 9, 9, 9, 9, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 9],
 [9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9],
 [9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9]]
```

```
XP = [0,1,0,-1]
YP = [-1,0,1,0]

for x in range(11):
    maze[0][x]= 1
    maze[8][x]= 1

for y in range(1,8):
    maze[y][0]= 1
    maze[y][10]=1

for y in range(1,8):
    for x in range(1,10):
        maze[y][x] = 0

for y in range(2,7,2):
    for x in range(2,7,2):
        maze[y][x]=1

for y in range(2,7,2):
    for x in range(2,9,2):
        d = random.randint(0,3)
        if x > 2:
            d = random.randint(0,2)
        maze[y+YP[d]][x + XP[d]] = 1
```

```
for y in range(27):
    for x in range(33):
        dungeon[y][x]=9

for y in range(1,8):
    for x in range(1,10):
        dx = x*3+1
        dy = y*3+1
        if maze[y][x] == 0:
            if random.randint(0,99)<20
                for ry in range(-1,2):
                    for rx in range(-1,2):
                        dungeon[dy+ry][dx+rx]=0
            else:
                dungeon[dy][dx]= 0
                if maze[y-1][x]==0:
                    dungeon[dy-1][dx] = 0
                if maze[y+1][x] == 0:
                    dungeon[dy+1][dx]=0
                if maze[y][x-1] == 0:
                    dungeon[dy][dx-1] = 0
                if maze[y][x+1] == 0:
                    dungeon[dy][dx+1] = 0
```

迷路をダンジョンに変える。

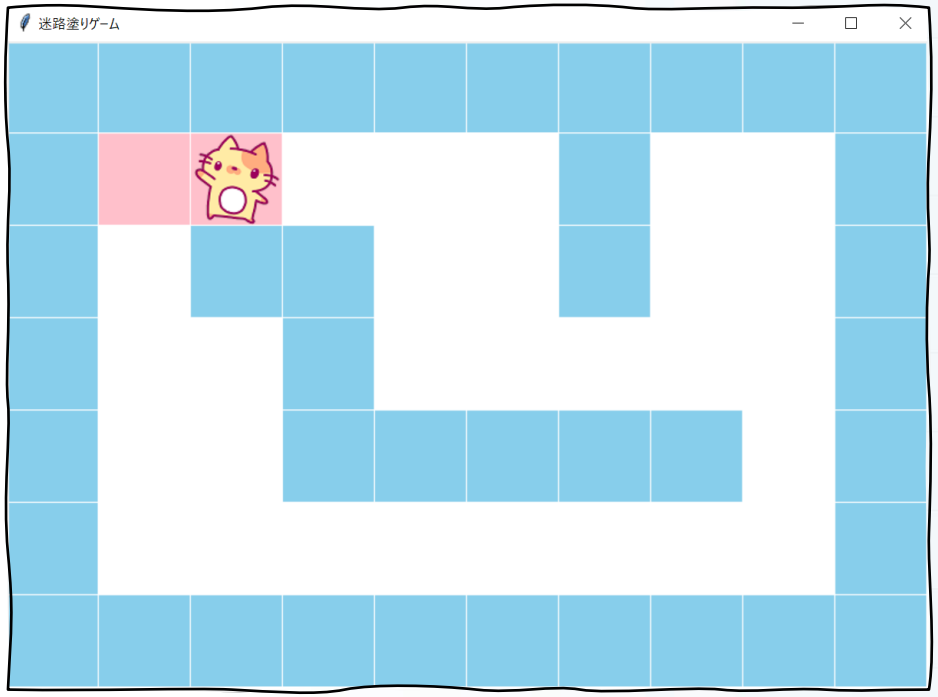
「Run Module」またはF5キーを押してプログラムを実行する。

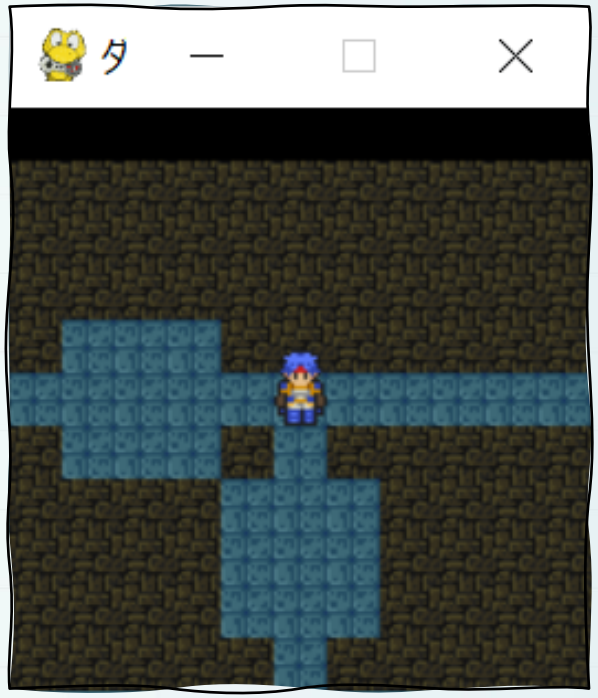


スペースキーを押すとダンジョンが再生成される。

ダンジョン内を移動する

ロールプレイングゲームはキャラクターをウィンドウの中央に表示し、方向キーの入力に応じて背景がスクロールしている。過去に作成したプログラムはキャラクタを方向キーで移動させていたが、今回は背景の方が移動するプログラムを作る。





上下左右キーを押すとキャラクタが移動する。背景は動かない。

上下左右キーを押すと背景が移動する。
※キャラクタは動いていない。

ダンジョン内を移動する

背景をスクロールさせるプログラムを作る。

```
import pygame
import sys
import random
#色の作成(RGBカラーで設定)
BLACK = (0,0,0)
#空っぽのリストを作成(迷路)
maze = []
for y in range(9):
    maze.append([0,0,0,0,0,0,0,0,0,0,0])
#空っぽのリストを作成(ダンジョン)
dungeon = []
for y in range(27):
    dungeon.append([0]*33)
```

次のページに続く

ダンジョン内を移動する

```
#画像ファイルのアップロード
imgWall = pygame.image.load("image/wall.png")
imgFloor = pygame.image.load("image/floor.png")
imgPlayer = pygame.image.load("image/player.png")
px = 4 #プレイヤのx座標(初期値)
py = 4 #プレイヤのy座標(初期値)
#ダンジョンを作る関数
#ダンジョンのもとになる迷路から作成
def make_dungeon():
    XP = [0,1,0,-1]
    YP = [-1,0,1,0]
    #迷路の周りの壁(一番上と一番下の壁)
    for x in range(11):
        maze[0][x]= 1
        maze[8][x]= 1
```

次のページに続く

ダンジョン内を移動する

```
#迷路の周りの壁(一番左と一番右の壁)
for y in range(1,8):
    maze[y][0]= 1
    maze[y][10]=1
#迷路の中を何もない状態に
for y in range(1,8):
    for x in range(1,10):
        maze[y][x] = 0
#迷路の柱を作る
for y in range(2,7,2):
    for x in range(2,7,2):
        maze[y][x]=1
#迷路の柱から上下左右に壁を作る
```

次のページに続く

ダンジョン内を移動する

```
for y in range(2,7,2):
    for x in range(2,9,2):
        d = random.randint(0,3)
        if x > 2:
            d = random.randint(0,2)
        maze[y+YP[d]][x + XP[d]] = 1
#上記で作成した迷路からダンジョンを作る
#全体を壁にする
for y in range(27):
    for x in range(33):
        dungeon[y][x]=9 #9は壁
#部屋と通路の配置
```

次のページに続く

ダンジョン内を移動する

```
for y in range(1,8):
    for x in range(1,10):
        dx = x*3+1 #ダンジョンのx座標
        dy = y*3+1 #ダンジョンのy座標
        if maze[y][x] == 0:
            if random.randint(0,99)<20: #部屋を作る(20%の確率で実行)
                for ry in range(-1,2): #ry:roomのy座標
                    for rx in range(-1,2): #rx:roomのx座標
                        dungeon[dy+ry][dx+rx]=0
            else: #通路を作る
                dungeon[dy][dx]= 0
                if maze[y-1][x]==0:
                    dungeon[dy-1][dx] = 0
                if maze[y+1][x] == 0:
                    dungeon[dy+1][dx]=0
```

次のページに続く

ダンジョン内を移動する

```
                    if maze[y][x-1] == 0:
                        dungeon[dy][dx-1] = 0
                    if maze[y][x+1] == 0:
                        dungeon[dy][dx+1] = 0
#ダンジョンを描画する
def draw_dungeon():
    screen = pygame.display.set_mode((176,176))
    screen.fill(BLACK)
    for y in range(-5,6):
        for x in range(-5,6):
            X=(x+5)*16
            Y=(y+5)*16
            dx = px+x #プレイヤの位置を基準にダンジョンを描画
            dy = py+y #プレイヤの位置を基準にダンジョンを描画
            if 0 <= dx and dx < 33 and 0<= dy and dy < 27:
```

次のページに続く

ダンジョン内を移動する

```
                    if dungeon[dy][dx] == 0:
                        screen.blit(imgFloor,[X,Y])
                    if dungeon[dy][dx] == 9:
                        screen.blit(imgWall,[X,Y])
                if x==0 and y==0:
                    screen.blit(imgPlayer,[X,Y-8])
#キャラクタ移動(主人公の位置をもとにダンジョンを描画させる)
def move_player():
    global px,py
    key = pygame.key.get_pressed()
    if key[pygame.K_UP]==1: #1:True 0:False
        if dungeon[py-1][px] !=9: #9は壁
            py=py-1
    if key[pygame.K_DOWN]==1: #1:True 0:False
        if dungeon[py+1][px] !=9: #9は壁
```

次のページに続く

ダンジョン内を移動する

```
                py=py+1
    if key[pygame.K_LEFT]==1: #1:True 0:False
        if dungeon[py][px-1] !=9: #9は壁
                px=px-1
    if key[pygame.K_RIGHT]==1: #1:True 0:False
        if dungeon[py][px+1] !=9: #9は壁
                px=px+1
def main():
    pygame.init()
    pygame.display.set_caption("ダンジョン内を歩く")
    clock = pygame.time.Clock()
    make_dungeon()
    while True:
        for event in pygame.event.get():
                if event.type == pygame.QUIT:
```

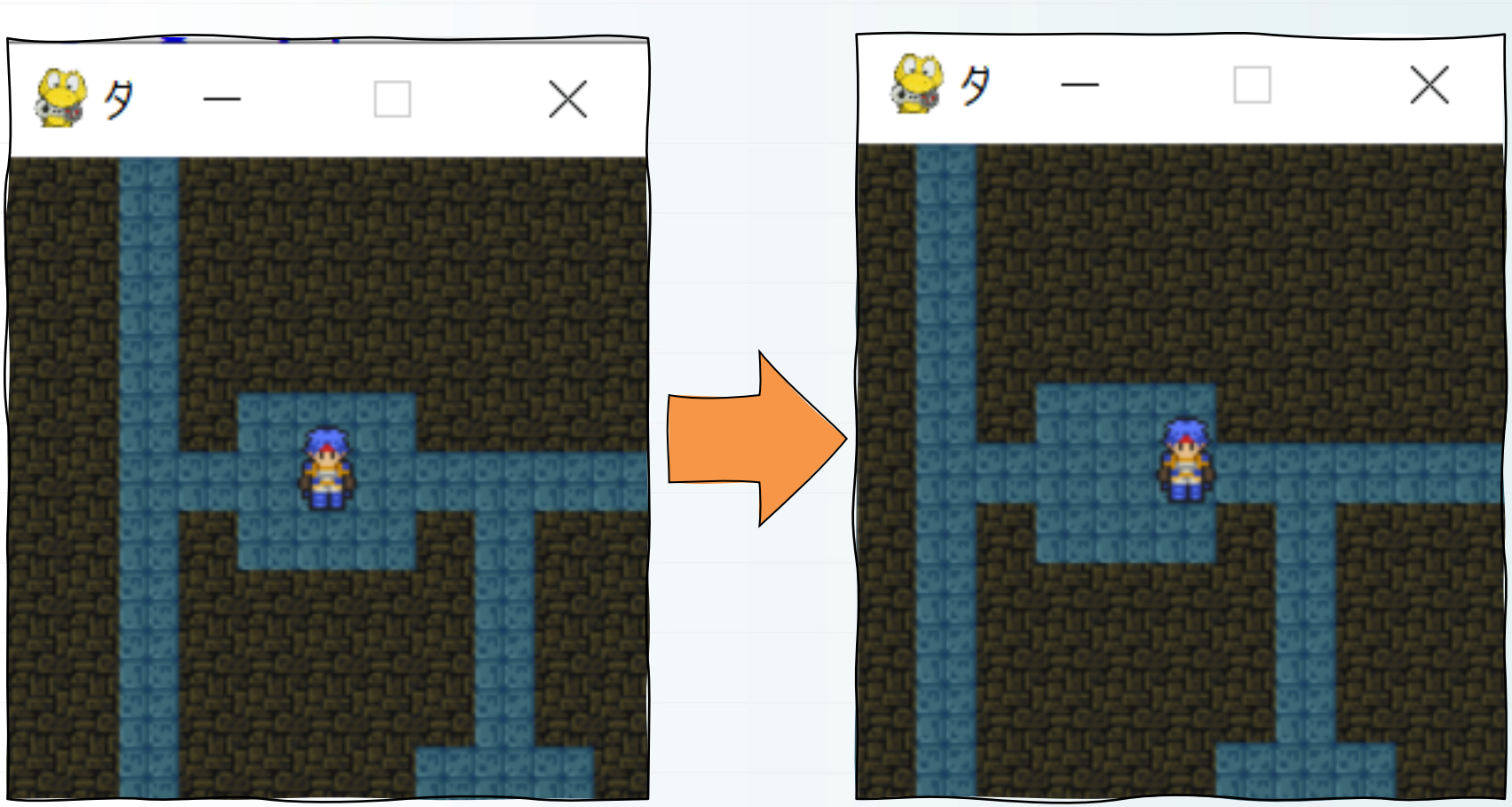
次のページに続く

ダンジョン内を移動する

```
                    pygame.quit()
                    sys.exit()
        move_player()
        draw_dungeon()
        pygame.display.update()
        clock.tick(5) #ミリ秒(1/1000 秒)単位


main()
```

ダンジョン内を移動する

「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：