もう，打ってみればいいじゃん

100行でゲーム

第1回　道路横断ゲームを作る

佐々木 弘隆

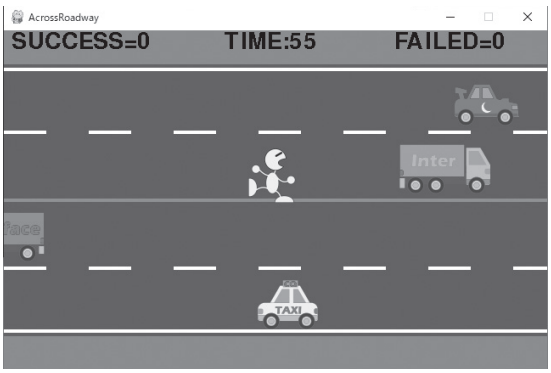


図1　今回作るゲームの完成形…全部で123行! できるかも

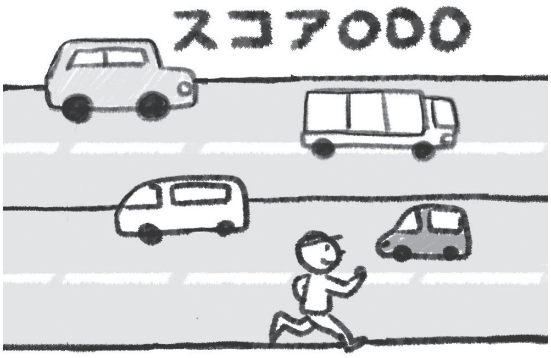


図2　今回作るゲームの画面イメージ
ラフなアイデア・スケッチでOK

趣味や仕事でプログラムを書いている人が大勢いる一方，授業でやらされたけど難しい，面白くないと思っている人も多いようです．それはとてももったいないことです．現代社会ではあらゆるモノがコンピュータで動いている，つまりプログラムが分かればできることはとても多いのです．

そこで，簡単なゲームを作成しながらプログラムをマスタしましょう．今回作るゲームの完成形を**図1**に示します．全部で123行のプログラムを書くだけなので，ちょっとした空き時間にチャレンジできます．分かりやすさや便利さで人気のプログラム言語Pythonを使っていきます．PythonやC言語などの経験があって，条件分岐や配列，ループなどを知っていると理解は早いですが，文法入門テキストを横に置きながら読み進めていっても大丈夫です．最低限の文法の理解は必要ですが，基本は習うより慣れろです．じゃんじゃんプログラムを書いて修正していきましょう．

ゲームの設計

早速一番楽しい設計であるゲーム・デザインをしていきましょう．

簡単に作れて触って面白いのはシンプルなアクション・ゲームでしょう．ゲームはストレス要因と成功報酬が必要です，ストレスばかりだとツラいですし，逆に報酬だけだと面白みがなくなります．

ゲームでは普段禁止されていることもできるので，いろいろな物事を題材にできる強みもあります．今回は身近な道路を題材にしましょう．車が走っている道路を横断するゲームです．ストレス要因は車にひかれることで，成功報酬が横断成功です．

4車線で移動方向と速度が違う車が動き回っていて，車にひかれないように人を歩かせます．スペース・キーを押すと一歩前進，後退はなしとします．

図2に今回作るゲームの画面をイメージしてみました．どんな画面にするのか考えてどんどんイメージとやる気を膨らませるのはとても大事なことです．十分にやる気が湧いた所でゲーム作成の方に進んでいきましょう．

環境構築

● Pythonのインストール

PythonはWindowsやLinux，macOSなどいろいろな環境に対応しています．詳しい設定方法や最新情報はPythonの公式を参照してください．

`https://www.python.org/`

と言っても特に難しいことはありません．

この記事ではWindowsを前提に進めていきますが，他の環境でも基本は変わりません．

プログラムは本誌サポート・ページから入手できます．
また，以下ページからもご覧いただけます．
https://interface.cqpub.co.jp/wp-content/uploads/AcrossRoadway.py_.pdf

表1　pygameの機能

機　能	説　明
画面処理	GUIウィンドウの制御など
図形描画	直線・四角・丸などの基本図形処理
画像描画	画像ファイルやイメージ処理
音声再生	音声ファイルから音楽と効果音再生
イベント	キーボードやマウスの読み取り

● pygameのインストール

また，この記事ではゲーム作成に便利なpygameというライブラリを使っていきます．

Pythonがインストールできたら，コマンド・プロンプト（各環境のCUI画面）で次のコマンドを実行してください．pygameライブラリが使えるようになります．

```
python -m pip install pygame
```

pygameには**表1**のような機能があります．

プログラミングするときに先に決めておくこと

● 課題のプログラムとゲーム・プログラムの違い

課題などで出されるプログラム[注1]の大半は上から下へ流れる1本道だと思います．これは「○○の処理をする」という目的が終わればプログラムも終了するという自然な流れです．

しかし，ゲームのプログラムは遊びたいと思う限りずっと続きます．そのため基本的に永遠に終わらないループ構造にしなくてはなりません．終わるときは「ゲーム・オーバになった」，「プレーヤがゲームを止めたい」が引き金になります．

● プログラムを関数化する

それとゲームに限らずプログラムの規模が少し大きくなってくると必要になるのが関数です．特定の目的を持った処理を関数と呼ばれる形式でまとめることでプログラム内のいろいろな場所から呼び出して利用できます．

プログラムを作っていて，似たような処理が増えてきたら関数にまとめていきましょう．

C言語やJAVAなどの他の言語で関数（ファンクションやメソッドとも呼ぶ）を触ったことがある人は概念は一緒ですので，言語による書式の違いだけ気を付けてください．

例えば，2つの数値を足し算する関数のPythonとC言語での記述は次のようになります．

• Python関数

```
def Add(a, b):
      return a + b
```

• C言語関数

```
int Add(int a, int b){
      return a + b;
}
```

また，PythonにはC言語などにおける始まりの処理であるmain関数は必須ではありません．ライブラリ宣言や関数を除いて，上の文から実行されます．

小さなプログラムの場合はこれでも問題ありませんが，プログラムが大きく複雑になると面倒な事態が発生する可能性があります．本連載のプログラムはかなり小さいのでmainを使わない方針でいきます．

● プログラムの文字コードはUTF-8で

私たちを含む非英語圏では文字データに独自の拡張をして自分たちの文字を使っています．英語しか想定していない環境で英語以外の文字（例えば日本語）を扱うと，いろいろな不具合が発生することが知られています．Pythonは一応，多言語に対応しているのですが，文字コードを指定しないと変な解釈をされてしまう場合があります．

Windowsは日本では伝統的にシフトJISが多用されていましたが，近年のプログラムではお勧めできません．プログラムを保存する時に文字コードを指定できるエディタが多い（メモ帳も*!*）ので，シフトJISからUTF-8に変更しておきます．最近は文字コードとしてUTF-8が無難です．

プログラムの先頭には使っている文字コードを宣言できるので1行目は次のように書いておきましょう．

```
# -*- coding: utf-8 -*-
```

● 使うライブラリ

このゲームでは3つのライブラリを使っていきます．

• ゲームに関係する処理をしてくれるpygame
• WindowsなどのOS関係の処理を取り持つsys
• ゲームに変化を付けるための乱数を生成するrandom

の3つです．この3つを使うためにプログラムの先頭に次のように書いておきます．

```
import pygame  # ゲーム用のライブラリ
import sys     # システム関連のライブラリ
import random  # 乱数関連のライブラリ
```

● 基本構造の実装

ゲーム本体を作る前に，その土台となる構造を作ります．

• pygameライブラリの初期化
• ウィンドウを作成します．

注1：一般的にPythonではスクリプトと呼びます．

リスト1　ゲーム・プログラムの基本構造

```
while (True):
    clock.tick(FPS)          # 更新間隔
    pygame.display.update() # 画面更新
    for event in pygame.event.get():              # イベント監視
        if event.type == pygame.KEYDOWN:          #   キーが押されてて
            elif event.key == pygame.K_ESCAPE:    #     ESCキーが押された時
                pygame.quit(); sys.exit()         #       終了処理
        elif event.type == pygame.QUIT:           #   終了イベントを受けた時
            pygame.quit(); sys.exit()             #     終了処理
```

この2つの手続きだけでアプリケーションのウィンドウが簡単に作成できてしまいます．

また，ゲームで時間の制御を行うためのクロックの用意もしておきます．

```
pygame.init()
screen = pygame.display.set_
             mode(SCREEN.size)
clock = pygame.time.Clock()
```

▶ゲームは無限ループ構造にする

次は処理の流れです，課題プログラムなどでは1本道で処理が終わったらプログラムも終わりますが，ゲームは終わらないのが基本です．ループ処理であるwhileの条件式を強制的に真（True）にして永遠にループをするようにします（**リスト1**）．こういう繰り返し処理のことを無限ループと呼びます．

終わらない構造を作るためには必須の方法ではありますが，それ以外の処理での無限ループは暴走につながる危険があるので特例と思ってください．

▶終了処理を追加する

もちろんこのゲームの無限ループもこのままですと暴走と変わりませんので終了処理を追加します．

pygameの機能を使ってイベントを取り込みます．Windowsなどではいろいろな動作をイベントとして処理します．イベントにはキーボードやマウスの操作やアプリケーション・ウィンドウ終了の［×］印のクリックも含まれます．

［×］印が押された時は終了イベントが届くのでアプリケーションを終了する処理を行います．pygame自体の終了をしてからWindowsにアプリケーション終了のイベントを送ります．

▶ESCキーで終了する処理を追加する

ここまででアプリケーション・ウィンドウの作成から［×］印を押しての終了までの構造ができました．

ただ，毎回［×］印をクリックするのも面倒なのでキーボードでの終了処理も追加します．キーが押されたイベントがあり，そのキーがESCキーだったときに終了することにしましょう．

最後に画面の更新と更新間隔を設定します．設定しないでいると使ってるPCの速度でゲームの速度が変わってしまうので遅すぎたり速すぎたりで滅茶苦茶になってしまうからです．これでゲーム・プログラムの基本構造が完成しました．

ゲームのプログラムを作る

● 基本構造の描画

基本構造の画面（**図3**）は真っ暗で寂しいのでゲームらしく彩りを加えていきます（**図4**）．

リスト2で最初はコンクリート・ジャングルらしく灰色で塗りつぶします．全体を塗りつぶすことで古いゲーム画面を消去して新しいゲーム画面の下地とします．

コンクリートの上にアスファルト，白線，破線，中

図3　基本構造の画面

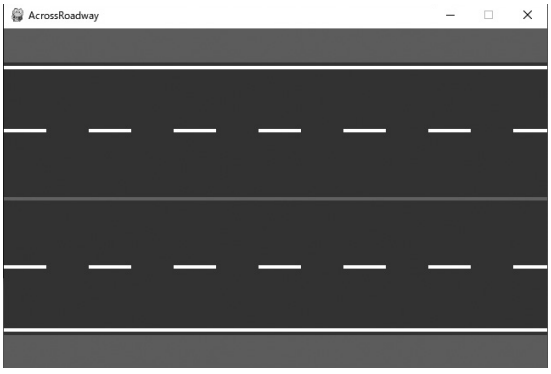


図4　コンクリートの上にアスファルト・白線・破線・中央分離帯を描いていく

リスト2　基本構造の画面を描画する

```
screen.fill("gray50") # 画面を灰色に塗り潰す
pygame.draw.rect(screen,"gray32",pygame.Rect(0,40,640,320))      # 車道のアスファルト
pygame.draw.rect(screen,"white",pygame.Rect(0,44,640,4))         # 上の歩道境界
pygame.draw.rect(screen,"white",pygame.Rect(0,352,640,4))        # 下の歩道境界
pygame.draw.rect(screen,"orangered2",pygame.Rect(0,198,640,4))  # 中央分離帯
for x in range(0, 7):    # 破線の描画
    pygame.draw.rect(screen,"white",pygame.Rect(100 * x,118,50,4)) #車線境界
    pygame.draw.rect(screen,"white",pygame.Rect(100 * x,278,50,4)) #車線境界
```

リスト3　画像ファイルからイメージ情報を読み込んでウィンドウに転送する

```
manSpr = pygame.image.load("man.png").convert_alp
ha()                                  # 人画像の読み込み
manRect = manSpr.get_rect()           # 人画像の矩形切り出し
manRect.center = (MAN_X, manY)        # 人画像の初期位置
screen.blit(manSpr, manRect)          # 人画像の描画
```

央分離帯を描いていきましょう．破線はループ処理で簡単に作れます．

● 色の指定

PCのように光を出すディスプレイの場合はRGBと呼ばれる方式で色を表すのが一般的です[注2]．Red，Green，Blueの三色の掛け合わせで表現します．Photoshopのようなグラフィック系ソフトやウェブ・ページのHTMLなどで見たことがある人も多いでしょう．

pygameでもRGBそれぞれの強さを0～255の範囲で設定して色を決めますが，色指定に慣れてない人は(64，255，12)のような数字を見ただけで色をイメージするのは難しいと思います．pygameではRGBの代わりに色の名前を直接指定することができます．色の名前と実際の色は下記URLを参照してください．

`https://www.pygame.org/docs/ref/color_list.html`

注2：他には印刷物のCMYKという概念があります．

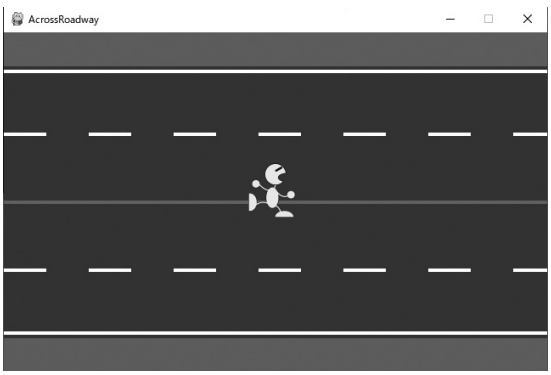


図5　人の移動

リスト4　音声の再生

```
seRun = pygame.mixer.Sound("run.mp3") # 足音音声の読み込み
seRun.play()                          # 足音の再生
```

● 画像の描画

四角などはpygameの基本図形機能で済みますが，車や人のような複雑な形状は基本図形の組み合わせでは大変なので画像を使います．

画像は自分で作れば何も問題無いのですが，作り方が分からない場合は誰かが作った画像を用意する方法があります．ただし，画像などの制作物には制作者作の権利が発生しています．権利者に有料か無料か，ゲームに使って大丈夫かを確認しましょう．

フリー素材で検索すると便利な素材が見つかりますが，印刷物やウェブ・サイトには使って大丈夫でもゲームには不可といったケースは結構あります．

このゲームの画像は筆者が制作した物です．記事を読んでいる方に対してゲームでも印刷物でも自由に使って良いという条件で公開します．

リスト3は画像ファイルからイメージ情報を読み込んで，画像の矩形(描画位置や横/縦幅など)の情報を用意してウィンドウに転送するプログラムです．

● 音声の再生

ゲームが無音だと盛り上がりに欠けるので，幾つかの音声を再生して楽しくしましょう．音声も画像と同じで権利問題があります．こちらも筆者が制作した物を画像と同じ条件で公開します．

音声は，足音，成功ジングル，失敗ブザーの3種類を用意しました．音声は音声ファイルから読み込んだあとは，好きなタイミングで再生ができます(**リスト4**)．

● 人の移動

スペースキーを押すと1回で一歩前進です(**図5**)．基本構造の終了処理のときにESCキーが押された判定をしていましたので，ここにスペース・キーの判定を追加します．また，前進する速度を設定します．

注意点として，1回で一歩進んだ後の座標(位置)へ

リスト5　人の移動

```
manY = manY - manSpeed / FPS     # 速度と制御間隔に応じて人のY座標を変える
manSpeed = max(0, manSpeed - manSpeed / FPS * 4)     # 人の減速（最低が0）
if manY <= MAN_END_Y:        # 上の歩道に到着したら
    manY = MAN_START_Y       # 人の初期座標
    manSpeed = 0             # 人の現在の速度
    success = success + 1    # 横断成功数を加算
    seSuccess.play()         # 横断成功音声を再生

for event in pygame.event.get():             # イベント監視
    if event.type == pygame.KEYDOWN:         #   キーが押されてて
        if event.key == pygame.K_SPACE:      #     スペースバーの時
            manSpeed = MAN_SPEED_MAX         #       人の移動開始
            seRun.play()                     #       足音の再生
        elif event.key == pygame.K_ESCAPE:   #     ESCキーが押された時
            pygame.quit(); sys.exit()        #       終了処理
```

リスト6　車の移動

```
LANE_LEFT = -100                          # 車道の左端座標
LANE_RIGHT = 740                          # 車道の右端座標
LANE_Y = [320,240,160,80]                 # 車線のY座標
LANE_SPEED_LOW = [-80,-40,40,100]         # 車線ごとの最低速度
LANE_SPEED_HIGH= [-150,-80,100,200]       # 車線ごとの最高速度

for i in range(0,len(LANE_SPEED_LOW)):       # 車線の分だけ車の処理をする
    carX[i] = carX[i] + carSpeed[i] / FPS        # 速度の応じた移動
    carRect[i].center = (carX[i], LANE_Y[i])     # 車の描画位置変更
    screen.blit(carSpr[i], carRect[i])           # 車の描画
    if LANE_SPEED_LOW[i] < 0:         # 左移動かつ
        if carX[i] <= LANE_LEFT:      # 左の端を過ぎたら
            carX[i] = LANE_RIGHT      # 右の出現位置に移動
            carSpeed[i] = random.uniform(LANE_SPEED_LOW[i],LANE_SPEED_HIGH[i])  # 移動速度も再計算
    else:                             # 右移動かつ
        if carX[i] >= LANE_RIGHT: # 右の端を過ぎたら
            carX[i] = LANE_LEFT  # 左の出現位置に移動
            carSpeed[i] = random.uniform(LANE_SPEED_LOW[i],LANE_SPEED_HIGH[i])  # 移動速度も再計算
```

の変更ではありません．それでは瞬間移動になってしまい不自然だからです．

そこで何回かに分けて指定した速度に従って移動をして，速度が0になると停止すると言うのがスムーズな移動を実現する方法です．

一歩前進なので，ここで足音を再生しておきましょう．ループの中で先程の速度に従った移動と減速処理を行います．道路を横断したら成功ジングルを再生してスコアを加算します．それと人を最初の位置に戻す処理で完了です（**リスト5**）．

● 車の移動

4車線分の車は自動的に動きます．車線のY座標・最低速度・最高速度をリスト構造であらかじめ作っておきましょう．リスト構造のデータを用意しておくと，ここを調整してゲームの面白さや難しさを調整しやすいです．

車は出現した時に最低速度から最高速度の範囲で乱数を使って速度が決められます．

同じ速度で移動し続けて，反対側に行くと最初の出現場所に戻って移動速度も新しい物が作成されます．

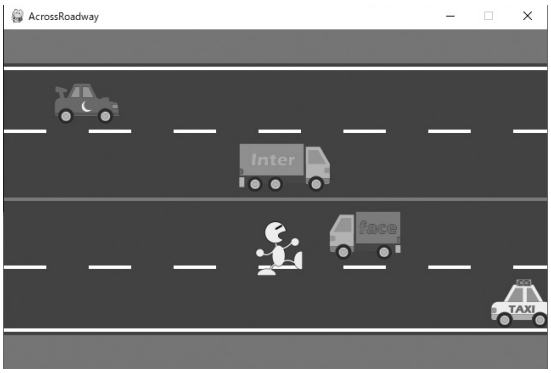


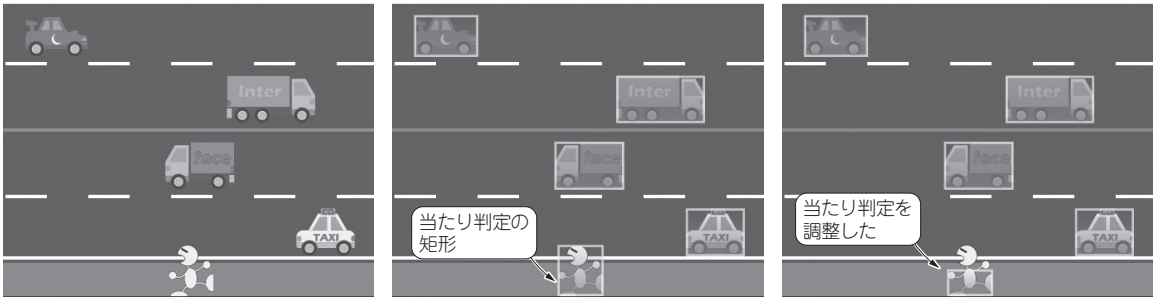
図6　車の移動

図6のようにだいぶゲームっぽい画面になってきました（**リスト6**）．

● 当たり判定

最後にストレス要因を作っていきましょう．

車にひかれると失敗ですから，人と車が接触しているかを判断する必要があります．これを当たり判定/



（a）ゲームの状況　（b）画像の矩形判定　（c）下半分の矩形を用意して当たり判定する

図7　当たり判定の修正

リスト7　当たり判定

```
manRect = manSprBody.get_rect()    # 人画像の矩形切り出し
manRect.center = (MAN_X, manY)     # 人画像の初期位置
manHitRect = pygame.Rect(MAN_X - manRect.
width/2,manY,manRect.width,manRect.height/2) # 当たり判定

if pygame.Rect.colliderect(manHitRect, carRect[i]):
                                        # 人との接触事故判定
    manY = MAN_START_Y  # 人の初期座標
    manSpeed = 0        # 人の現在の速度
    failed = failed + 1 # 横断失敗数を加算
    seFalse.play()      # 横断失敗音声を再生
```

リスト8　人を最初の位置に戻す処理を関数化する

```
def ManStartPos():          # 人をスタート位置にセットする関数
    global manY, manSpeed # グローバル変数へアクセス宣言
    manY = MAN_START_Y      # 人の初期座標
    manSpeed = 0            # 人の現在の速度

if pygame.Rect.colliderect(manHitRect, carRect[i]):
                                        # 人との接触事故判定
    ManStartPos()       # 人を初期位置に
    failed = failed + 1 # 横断失敗数を加算
    seFalse.play()      # 横断失敗音声を再生
```

リスト9　スコアの表示

```
sysfont = pygame.font.SysFont(None, 40)  # 標準フォント
                                                指定
success = failed = 0                # 横断数と失敗数の初期化
screen.blit(sysfont.render("SUCCESS="+str(success),
False, (0,0,0)), (10,0)) # 横断成功数の描画
screen.blit(sysfont.render("FAILED="+str(failed),
False, (0,0,0)), (460,0))  # 横断失敗数の描画
```

ヒット・チェック/コリジョン・チェックなどと呼びます（ここでは当たり判定と呼ぶ）.

当たり判定は若干面倒な処理なのですが，pygameの機能に矩形の当たり判定が用意されています．矩形は画像の描画のときに既に用意しているので利用できそうです.

ただし少々問題があって，このままだと画像の大きさそのままの判定になってしまいます.

図7（a）の状況ではプレーヤは車の手前に人がいるだけなのでひかれないと人間は判断しますが，**図7**（b）の画像の矩形判定ではひかれてしまいます.

そこで**図7**（c）のように人の下半分の矩形を用意して，車との当たり判定に使うと自然な判定ができます．車にひかれたら失敗ブザーを再生して，失敗スコアを加算します．人も最初の位置に戻しましょう（**リスト7**）.

● 関数化

横断成功のときも車にひかれたときも，人を最初の位置に戻すことをしています．こういう共通処理は関数にしておくとスッキリ見やすく，ミスも少なくなります（**リスト8**）.

● スコアの表示

最後にゲームのスコアを表示します.

画面に描画するにはCUIで使ったprintの代わりにrenderで文字列を画像にレンダリングしてblitで画面に画像を描画します（**リスト9**）.

これで冒頭の**図1**のゲームの完成です．ここまでのリストを1つにまとめたものと画像・音声データはダウンロード・ページからダウンロードできます.

ゲームの完成，そして

これにて一応完成しました．一応と言うのがチョットもやっとしますが理由があります．ゲームを面白くする，という大事な工程が残っているからです．そのためにゲーム内容の調整がここから始まります．人の速度をもっと早くしたらどうか？車線ごとの速度のメリハリを極端にしたらどうか？スコアの位置は？車のデザインは？などなど….

絶対的な正解はありませんので自分なりの正解を目指してプログラムをいじり回してみましょう.

ささき・ひろたか