“DS/AI 프로그래밍”

9주차 PyGame (블록깨기 게임 구현)

J.-K.- Seo
데이터과학원

Chapter

05 블록 깨기

설명이 필요 없는 유명한 게임입니다. 화면 아래에 있는 패들을 조작해서 공을 튕겨서 화면 위에 있는 블록을 전부 제거하면 됩니다.

소스 코드(blocks.py)

```
""" blocks.py - Copyright 2016 Kenichiro Tanaka """
import sys
import math
import random
import pygame
from pygame.locals import QUIT, KEYDOWN, K_LEFT, K_RIGHT, Rect

class Block:
    """ 블록, 공, 패들 객체 """
    def __init__(self, col, rect, speed=0):
        self.col = col
        self.rect = rect
        self.speed = speed
        self.dir = random.randint(-45, 45) + 270

    def move(self):
        """ 공을 움직인다 """
        self.rect.centerx += math.cos(math.radians(self.dir))\
            * self.speed
        self.rect.centery -= math.sin(math.radians(self.dir))\
            * self.speed

    def draw(self):
        """ 블록, 공, 패들을 그린다 """
        if self.speed == 0:
            pygame.draw.rect(SURFACE, self.col, self.rect)
        else:
            pygame.draw.ellipse(SURFACE, self.col, self.rect)

def tick():
    """ 프레임별 처리 """
    global BLOCKS
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
        elif event.type == KEYDOWN:
```

```
            if event.key == K_LEFT:
                PADDLE.rect.centerx -= 10
            elif event.key == K_RIGHT:
                PADDLE.rect.centerx += 10
    if BALL.rect.centery < 1000:
        BALL.move()

    # 블록과 충돌?
    prevlen = len(BLOCKS)
    BLOCKS = [x for x in BLOCKS
              if not x.rect.colliderect(BALL.rect)]
    if len(BLOCKS) != prevlen:
        BALL.dir *= -1

    # 패들과 충돌?
    if PADDLE.rect.colliderect(BALL.rect):
        BALL.dir = 90 + (PADDLE.rect.centerx - BALL.rect.centerx) \
            / PADDLE.rect.width * 80

    # 벽과 충돌?
    if BALL.rect.centerx < 0 or BALL.rect.centerx > 600:
        BALL.dir = 180 - BALL.dir
    if BALL.rect.centery < 0:
        BALL.dir = -BALL.dir
        BALL.speed = 15

pygame.init()
pygame.key.set_repeat(5, 5)
SURFACE = pygame.display.set_mode((600, 800))
FPSCLOCK = pygame.time.Clock()
BLOCKS = []
PADDLE = Block((242, 242, 0), Rect(300, 700, 100, 30))
BALL = Block((242, 242, 0), Rect(300, 400, 20, 20), 10)

def main():
    """ 메인 루틴 """
    myfont = pygame.font.SysFont(None, 80)
    mess_clear = myfont.render("Cleared!", True, (255, 255, 0))
    mess_over = myfont.render("Game Over!", True, (255, 255, 0))
```

```
    fps = 30
    colors = [(255, 0, 0), (255, 165, 0), (242, 242, 0),
              (0, 128, 0), (128, 0, 128), (0, 0, 250)]

    for ypos, color in enumerate(colors, start=0):
        for xpos in range(0, 5):
            BLOCKS.append(Block(color,
                    Rect(xpos * 100 + 60, ypos * 50 + 40, 80, 30)))

    while True:
        tick()

        SURFACE.fill((0, 0, 0))
        BALL.draw()
        PADDLE.draw()
        for block in BLOCKS:
            block.draw()

        if len(BLOCKS) == 0:
            SURFACE.blit(mess_clear, (200, 400))
        if BALL.rect.centery > 800 and len(BLOCKS) > 0:
            SURFACE.blit(mess_over, (150, 400))

        pygame.display.update()
        FPSCLOCK.tick(fps)

if __name__ == '__main__':
    main()
```

1 : 개요

화면 위에 그려진 요소는 패들, 블록, 공 세 가지입니다. 얼핏 보면 다른 것 같지만 단색으로 빈틈없이 칠하는 일정 영역이 있다는 공통점이 있습니다.

	패들	블록	공
그리기	직사각형. 칠하는 영역은 Rect로 지정	직사각형. 칠하는 영역은 Rect로 지정	원. 칠하는 영역은 Rect로 지정
동작	사용자가 움직인다	움직이지 않는다	자동으로 움직인다

공은 원형으로 스스로 이동하기 때문에 방향과 속도 정보가 필요합니다. 그래서 속도 프로퍼티를 갖게 해서 속도가 1이면 직사각형, 그렇지 않으면 원형으로 구별하게 했습니다.

3가지를 다른 클래스로 구현해도 됩니다. 이번은 코드양을 줄이려고 하나의 클래스로 구현했습니다.

2 : 전역 변수

이번 게임에서는 다음 전역 변수를 사용합니다.

BLOCKS	블록 객체를 저장하는 리스트
PADDLE	패들 객체(Block 클래스의 인스턴스)
BALL	공 객체(Block 클래스의 인스턴스)

또한, SURFACE(윈도)와 FPSCLOCK(프레임 레이트 조정용의 타이머) 변수를 사용합니다.

3 : 클래스와 함수

Block 클래스

프로퍼티

col	채우는 색
rect	그리는 직사각형(위치와 크기)

speed	이동 속도. 공만 기본값 0
dir	이동 방향(단위: 도) 공만 사용

메서드

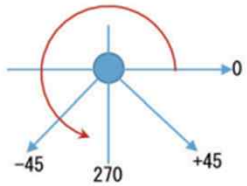
move	공을 움직인다
draw	그린다(공: 원, 기타: 직사각형)

프로퍼티는 생성자에서 초기화합니다.

```
def __init__(self, col, rect, speed=0):
```

speed=0으로 기본값을 지정했습니다. 공의 객체를 작성할 때는 속도 speed를 지정합니다. 블록과 패들은 속도를 지정하지 않습니다. 공의 발사 각도는 270도를 중심으로 해 ±45도가 되도록 난수로 정합니다.

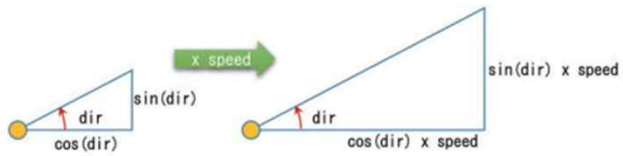
```
self.dir = random.randint(-45, 45) + 270
```



공의 이동은 move 메서드에서 실시합니다.

```
        self.rect.centerx += math.cos(math.radians(self.dir))\
            * self.speed
        self.rect.centery -= math.sin(math.radians(self.dir))\
            * self.speed
```

방향 dir을 함수 math.radians()를 사용해 라디안으로 변환하고, 그 값을 cos/sin에 넘겨서 X축 방향과 Y축 방향의 성분을 구하며, 마지막으로 speed를 곱해서 실제의 이동량을 구합니다.



PC 화면에서는 Y 축은 아래 방향이 양수이므로, centery에 값을 더할 때 거꾸로 맞추는 것에 주의하세요.

그리기는 draw 메서드에서 실시하는데, speed가 0인지 아닌지 여부로 직사각형이나 원을 전환합니다. 지정된 곳 rect에 지정된 색 col로 단순히 그릴 뿐입니다.

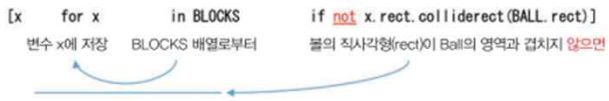
tick()

매 프레임마다 호출되는 함수입니다. pygame 이벤트를 꺼내고, QUIT이면 게임을 종료합니다. 이벤트 종류 type이 KEYDOWN일 때, 그 key가 K_LEFT라면 패들의 X 좌표를 −10, K_RIGHT라면 +10 합니다. 이로써 좌우 키를 누르면 패들이 이동합니다. 다음에 BALL.move()로 공을 이동합니다.

이동한 뒤에는 충돌 판정을 합니다. 먼저 블록과의 충돌입니다. 처음에 충돌 전의 블록을 셉니다. 다음에 볼과 충돌한 블록을 리스트에서 삭제합니다. 마지막으로 충돌 후의 블록을 셉니다. 충돌 전과 충돌 후의 블록 수가 다르다면 「공이 블록에 맞은 것」이므로 공 방향을 바꿉니다. 그 부분의 코드는 다음과 같습니다.

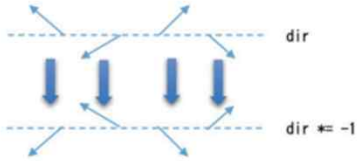
```
prevlen = len(BLOCKS)
BLOCKS = [x for x in BLOCKS
          if not x.rect.colliderect(BALL.rect)]
if len(BLOCKS) != prevlen:
    BALL.dir *= -1
```

충돌 전의 블록 수를 len(BLOCKS)로 구하고, 변수 prevlen에 저장합니다. 충돌한 블록을 삭제하는 처리는 리스트 내포 표기를 사용해 기술합니다.



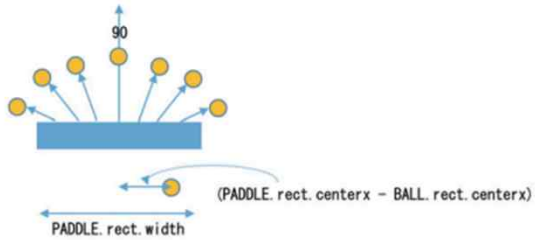
Rect 클래스의 colliderect 메서드를 사용해 2개 영역이 중복되는지를 검출합니다. 이처럼 리스트 내포 표기를 사용하면 'BLOCKS 배열 안에서 BALL과 겹치지 않는 요소로 구성된 리스트를 반환'하는 처리를 간단하게 기술합니다.

볼 방향을 바꾸는 것은 BALL.dir *= -1 처리입니다.



예를 들어, 45도 각도로 나가는 공이 -45도로 방향을 바꾸는 상황을 생각하면 -1을 곱하는 의미를 알 것이라 생각합니다.

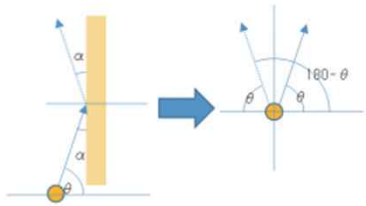
다음은 패들과의 충돌입니다. 패들과의 충돌도 Rect의 colliderect 메서드를 사용합니다. 패들에 부딪혀 반사할 때는 충돌 장소에 따라 반사각이 변하도록 조정합니다. 윗 방향이 90도입니다. 패들 중심과 공 중심의 거리를 구하고 그 값을 패들 폭으로 나눕니다. 그 값에 상수(임시로 80)를 곱해서 중심 90을 더합니다.



좌우 벽과의 충돌 판정·반사용 코드는 다음과 같습니다. x축 값이 0 미만이나, 600보다 크면 충돌로 간주합니다.

```
if BALL.rect.centerx < 0 or BALL.rect.centerx > 600:
    BALL.dir = 180 - BALL.dir
```

왜 180 − BALL.dir 식으로 반사각을 계산하는지 다음 그림을 통해 설명합니다.



좌우 그림을 보면 벽에 입사하는 각도 α와 반사각 α는 같아야만 합니다. 다만, 실제로 공의 움직이는 방향은 자신이 중심이 되므로, α가 아닌 θ입니다. 반사 후의 공 방향은 오른쪽 위 그림과 같이 「180−θ」입니다. 입사각과 반사각이 아닌 공이 향하는 방향을 변화시켜야 해야 하는 것에 주의하세요.

위의 벽과의 충돌 판정과 반사는 다음 코드입니다. Y 축 값이 0 미만이면 충돌로 합니다. 속도를 가속하고 있는 것을 알 수 있습니다.

```
if BALL.rect.centery < 0:
    BALL.dir = -BALL.dir
    BALL.speed = 15
```

main()

먼저 폰트를 만들고 클리어 시와 게임 오버 시의 비트맵을 작성합니다. 변수 fps(frame per second: 1초 동안의 프레임 수)와 colors를 초기화합니다. 블록을 배치하는 건 다음 부분입니다.

```
for ypos, color in enumerate(colors, start=0):
    for xpos in range(0, 5):
        BLOCKS.append(Block(color,
                Rect(xpos * 100 + 60, ypos * 50 + 40, 80, 30)))
```

색을 차례로 꺼내고, 블록 장소를 계산하기 위한 번호도 필요하기 때문에 enumerate 함수를 사용했습니다. 이것은 이터러블 객체를 반환하는 함수로 번호(0부터 시작하는 요소의 순서)와 요소로 구성된 튜플을 되돌려줍니다.

```
colors = [(255, 0, 0), (255, 165, 0), (242, 242, 0), (0, 128, 0), (128, 0, 128), (0, 0, 250)]
for ypos, color in enumerate(colors, start=0):
```

ypos가 세로 방향의 번호, xpos가 가로 방향의 번호입니다. 이러한 값을 사용해서 X와 Y 좌표의 위치를 계산하고 블록 객체를 만들고 배열 BLOCKS에 추가하는 것이 다음 줄입니다.

```
BLOCKS.append(Block(color,
                    Rect(xpos * 100 + 60, ypos * 50 + 40, 80, 30)))
```

나머지는 while 문으로 메인 루프에 진입합니다. tick()에서 프레임별 처리를 하고 SURFACE.fill((0, 0, 0))으로 화면을 검정으로 칠하고 공, 패들, 블록을 그립니다.

리스트 BLOCKS의 길이가 0이 되었을 때, 모든 블록을 지웠다는 뜻으로 클리어 메시지를 표시합니다. 반대로 공의 Y 좌표가 800보다 커질 때는 공이 화면 아래를 지나간 것으로 게임 오버 메시지를 표시합니다.

마지막으로 pygame.display.update()로 그린 것을 화면에 반영하고 FPSCLOCK.tick(fps)로 일정 FPS 수가 되도록 조정합니다.

게임 설명은 이상입니다. 대략 100줄로 채우겠다는 목표를 아슬아슬하게 달성했습니다. 게임으로서의 완성도는 아직 멀었습니다. 점수를 추가하거나 여러 스테이지를 준비하거나 공 갯수를 여러 개로 하거나 추가할 사항은 많습니다. 꼭 높은 품질의 오리지널 블록 깨기 게임을 만들어 보세요.

KUIDS
고려대학교 데이터과학원

The End