38장

진급 프로젝트:태그 게임

38장 진급 프로젝트:태그 게임

38.1 문제 구성 요소 찾기

38.2 창에 두 가지 모양 만들기

38.3 캔버스 안에서 모양 움직이기

38.4 두 모양 사이의 충돌 검출하기

38.5 개선할 수 있는 부분

38.6 요약

38 프로젝트 문제

» **tkinter 라이브러리를 사용하는 GUI 게임을 작성하라. 태그(Tag)라는 게임을 시뮬레이션하는 게임이다.**

- 창 안에 플레이어를 두 명 만든다. 시작할 때 플레이어의 크기와 위치는 난수로 결정한다.
- 두 플레이어는 키보드 하나를 사용한다. 한 플레이어는 W, A, S, D를 사용하고, 다른 플레이어는 I, J, K, L을 사용해 자신의 말을 움직인다.
- 두 플레이어는 창 안에서 자기 말을 키보드로 이동시키면서 상대편을 잡기 위해 노력한다.
- 두 플레이어 말이 서로 부딪치면 '태그!'라는 단어가 화면에 나타나야 한다.

» **이 게임은 단순하며, 코드도 그리 길지 않다. 게임과 같은 GUI나 시각적인 애플리케이션을 작성 할 때는 처음에 너무 원대한 목표를 잡지 않는 것이 중요하다. 단순한 문제부터 시작해 모든 것이 잘 작동한다는 사실을 확인해 나가면서 점점 프로그램을 키워 나가라.**

38.1 문제 구성 요소 찾기

38.1 문제 구성 요소 찾기

» **문제가 주어지면 그 문제를 구성하는 요소들을 알아내야 한다. 코드를 점차 발전시켜 나가면 코드 작성이 훨씬 더 쉽다. 이 문제의 경우 궁극적으로 다음 세 가지 목표를 달성해야 한다.**

- 두 가지 모양을 만든다.
- 앞에서 만든 모양을 눌린 키에 따라 이동시킨다.
- 두 모양이 서로 접촉했는지 판단한다.

38.2 창에 두 가지 모양 만들기

38.2 창에 두 가지 모양 만들기

» **지금까지 본 다른 GUI와 마찬가지로, 첫 번째 단계에서는 게임에 쓸 창을 만들고 필요한 위젯을 추가한다. 코드 38-1을 보자.**

- 창에는 캔버스 위젯만 들어 있다. 캔버스 위젯은 모양이나 다른 그래픽 요소들을 그릴 수 있는 직사각형 영역이다.

```
import tkinter
window = tkinter.Tk()
window.geometry("800x800")
window.title("태그!")
canvas = tkinter.Canvas(window) ---- 플레이어의 말을 그릴 캔버스 위젯
canvas.pack(expand=1, fill='both') ---- 창 크기가 변해도 그에 맞춰 캔버스가 창 전체를
                                        차지하도록 캔버스 위치를 지정
```

코드 38-1
창과 위젯 초기화하기

38.2 창에 두 가지 모양 만들기

» **이제 각 플레이어를 위한 모양을 만든다. 플레이어의 말을 직사각형으로 만들 것이다. 별도의 위젯이 아니라 캔버스에 더해지는 객체로 각각의 모양을 만든다.**

- 플레이어가 두 명 이상이기 때문에 이를 좀 더 모듈화할 수 있게 생각하는 것이 좋다.
- 플레이어를 표현하는 클래스를 만들고, 그 클래스에 속한 객체가 캔버스에 표시할 플레이어의 말을 초기화하게 하자.

38.2 창에 두 가지 모양 만들기

그림 38-1은 직사각형(수학적으로는 정사각형)을 만드는 방법을 나타낸 것이다.

- 좌상단(왼쪽 위) 꼭지점 좌표 x1, y1을 선택한다. 그리고 직사각형의 크기를 선택한다. 우하단(오른쪽 아래) 꼭지점 좌표인 x2, y2는x1, y1과 크기를 가지고 계산한다.

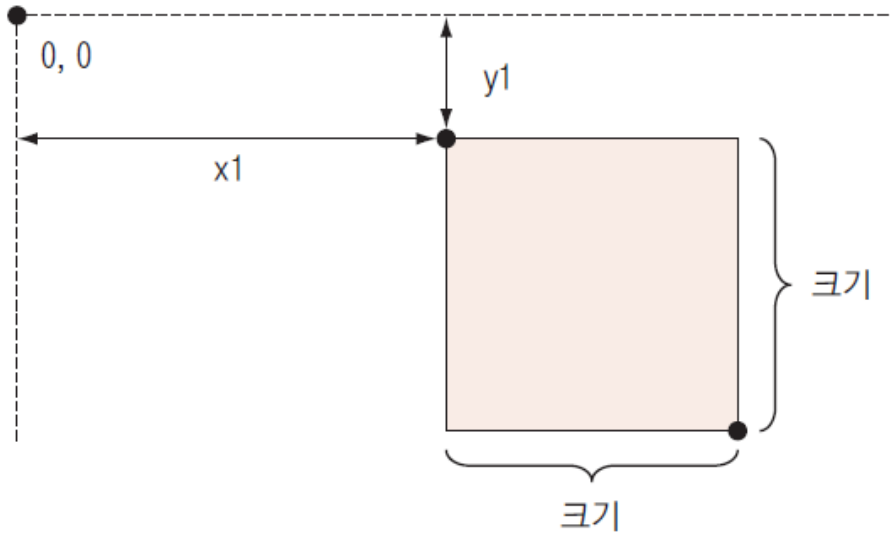

그림 38-1
좌상단 좌표와 크기를 난수로 선택함으로써 직사각형으로 된 플레이어의 말을 구성

38.2 창에 두 가지 모양 만들기

코드 38-2는 플레이어의 직사각형 말을 만드는 코드다.

- 이 코드는 Player라는 클래스를 만든다.
- 직사각형을 사용해 플레이어의 말을 표현한다. 캔버스에 있는 객체는 x1, y1, x2, y2라는 네 가지 정수로 표현할 수 있다. (x1,y1)은 좌상단 좌표, (x2,y2)는 우하단 좌표다.

```
import random
class Player(object):
    def __init__(self, canvas, color):
        self.color = color
        size = random.randint(1,100)
        x1 = random.randint(100,700)
        y1 = random.randint(100,700)
        x2 = x1+size
        y2 = y1+size
        self.coords = [x1, y1, x2, y2]
        self.piece = canvas.create_rectangle(self.coords)
        canvas.itemconfig(self.piece, fill=color)
```

모양을 더할 캔버스와 모양을 표현할 색을 데이터 속성으로 저장하는 객체를 위한 초기화 메서드

객체의 데이터 속성으로 color 설정

1 이상 100 미만의 정수를 플레이어 말 크기로 정함

플레이어 말의 좌상단 좌표의 x값을 정해진 범위에서 선택

플레이어 말의 좌상단 좌표의 y값을 정해진 범위에서 선택

플레이어 말의 우하단 좌표의 x값

플레이어 말의 우하단 좌표의 y값

모양을 표시할 좌표를 리스트로 만들고 객체의 데이터 속성으로 저장

플레이어의 말을 캔버스상의 직사각형으로 설정. 이때 앞에서 계산한 좌표를 사용

앞 줄에서 모양을 저장한 self.piece라는 변수를 사용해 플레이어의 말을 어떤 색으로 표시할지 지정함

코드 38-1 플레이어 클래스

38.2 창에 두 가지 모양 만들기

창을 만들고 다음 코드로 플레이어들을 캔버스에 추가한다.

- 이 코드는 같은 캔버스 위에 두Player 타입 객체를 만든다. 하나는 노란색, 다른 하나는 파란색이다.

```
player1 = Player(canvas, "yellow")
player2 = Player(canvas, "blue")
```

38.2 창에 두 가지 모양 만들기

이 코드를 실행하면 그림 38-2와 비슷한 창을 볼 수 있다.

- 임의의 위치에 크기가 무작위로 정해진 두 정사각형이 보일 것이다. 마우스를 클릭하거나 키보드의 키를 눌러도 아무 일도 발생하지 않을 것이다.

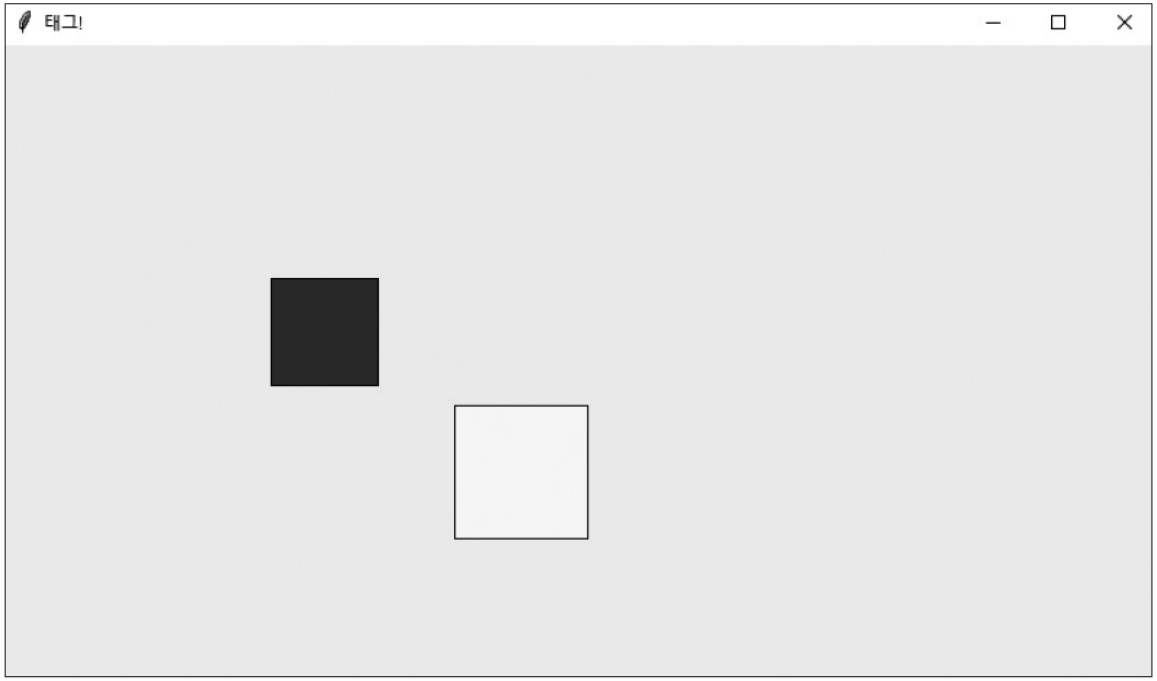

그림 38-2
두 플레이어 객체를 만든 게임 화면. 두 정사각형의 크기와 위치는 프로그램을 실행할 때마다 달라진다

38.3 캔버스 안에서 모양 움직이기

38.3 캔버스 안에서 모양 움직이기

모양은 키 눌림이라는 유형의 이벤트에 대해 반응한다.

- 모양을 위로 움직이려면 한 플레이어는 W, 다른 플레이어는 I를 눌러야 한다.
- 모양을 왼쪽으로 움직이려면 한 플레이어는 A, 다른 플레이어는 J를 눌러야 한다.
- 모양을 오른쪽으로 움직이려면 한 플레이어는 S, 다른 플레이어는 K를 눌러야 한다.
- 모양을 아래로 움직이려면 한 플레이어는 D, 다른 플레이어는 L을 눌러야 한다.

캔버스의 키 눌림 이벤트를 처리하는 함수를 만든다. 그 함수 안에서 눌린 키에 따라서 두 플레이어 중 한쪽을 움직여야 한다.

38.3 캔버스 안에서 모양 움직이기

» 이 코드에서 move는 Player 클래스 안에 정의된 함수다. 그 함수는 인자가 "u"면 위, "d"면 아래, "r"이면 오른쪽, "l"이면 왼쪽으로 플레이어를 움직인다.

```
def handle_key(event): ---- 이벤트 핸들러 함수
    if event.char == 'w' : ---- 이벤트에서 눌린 키가 W인지 검사
        player1.move("u") ---- move는 Player 클래스에 정의된 메서드. 그 메서드를 호출해서 모양을 위로 이동
    if event.char == 's' :
        player1.move("d")
    if event.char == 'a' :
        player1.move("l")
    if event.char == 'd' :
        player1.move("r")
    if event.char == 'i' : ---- 이벤트에서 눌린 키가 I인지 검사
        player2.move("u") ---- move는 Player 클래스에 정의된 메서드.
                              다른 플레이어에 대해 그 메서드를 호출해서 모양을 위로 이동
    if event.char == 'k' :
        player2.move("d")
    if event.char == 'j' :
        player2.move("l")
    if event.char == 'l' :
        player2.move("r")
```

38.3 캔버스 안에서 모양 움직이기


```
window = tkinter.Tk()
window.geometry("800x800")
window.title("태그!")
canvas = tkinter.Canvas(window)
canvas.pack(expand=1, fill='both')

player1 = Player(canvas, "yellow")
player2 = Player(canvas, "blue")
canvas.bind_all('<Key>', handle_key) ---- 캔버스 키 눌림 이벤트는 모두 handle_key 함수가 처리

window.mainloop()
```

코드 38-3
캔버스에서 키가 눌린 경우를 처리하는 이벤트 핸들러 함수

38.3 캔버스 안에서 모양 움직이기

Player 클래스 안에 좌표 값을 바꿔서 모양을 옮기는 코드를 작성할 수 있다. 코드 38-4를 보자.

- 플레이어가 움직일 수 있는 방향에 따라 coords 데이터 속성 값을 변경한 다음, self.piece 변수와 self.coords 변수를 캔버스의 coords()메서드에 넘겨서 플레이어 말의 위치를 변경한다.
- 이벤트가 한 번에 하나씩 이벤트 핸들러에 전달되므로 한 번에 한 사용자의 키 입력만 처리할 수 있다.

38.3 캔버스 안에서 모양 움직이기


```
class Player(object):
    def __init__(self, canvas, color):
        size = random.randint(1,100)
        x1 = random.randint(100,700)
        y1 = random.randint(100,700)
        x2 = x1+size
        y2 = y1+size
        self.color = color
        self.coords = [x1, y1, x2, y2]
        self.piece = canvas.create_rectangle(self.coords, tags=color)
        canvas.itemconfig(self.piece, fill=color)

    def move(self, direction): ---- 'u', 'd', 'l', 'r' 방향 중 한쪽으로 모양을 움직이는 메서드
        if direction == 'u': ---- 네 가지 가능한 입력('u', 'd', 'l', 'r') 중 하나인 경우 입력에 따라 다른 동작 수행
            self.coords[1] -= 10 ----
            self.coords[3] -= 10 ---- 위로 움직일 경우 coords 리스트의 y1과 y2 값을 10 감소시킴
            canvas.coords(self.piece, self.coords) ---- self.piece가 가리키는 직사각형의 좌표 변경
```

38.3 캔버스 안에서 모양 움직이기


```
        if direction == 'd':
            self.coords[1] += 10
            self.coords[3] += 10
            canvas.coords(self.piece, self.coords)
        if direction == 'l':
            self.coords[0] -= 10
            self.coords[2] -= 10
            canvas.coords(self.piece, self.coords)
        if direction == 'r':
            self.coords[0] += 10
            self.coords[2] += 10
            canvas.coords(self.piece, self.coords)
```

코드 38-4
캔버스 안에서 모양 움직이기

38.3 캔버스 안에서 모양 움직이기

» 이제 프로그램을 실행하면 W, A, S, D 키는 노란 사각형을 움직이고, I, J, K, L 키는 파란 사각형을 움직일 것이다.

- 키 이벤트 특성상, 어떤 키를 계속 누르고 있으면 그 키가 뜻하는 방향으로 모양이 계속 움직임을 알 수있다.
- 하지만 키를 누르고 있는 중간에 다른 키를 누르면 움직임이 멈추고 다른 키에 맞는 새로운 움직임이 시작된다(그리고 그 키에서 손을 떼거나 또 다른 키가 눌릴 때까지 움직임이 계속된다).

38.4 두 모양 사이의 충돌 검출하기

38.4 두 모양 사이의 충돌 검출하기

» 이 게임은 태그 게임이므로, 한 모양이 다른 모양과 서로 부딪쳤는지 알 수 있으면 좋을 것이다.

- 코드 로직은 캔버스 위젯 객체에 대해 두 가지 메서드를 호출하는 것으로 이뤄진다.
- 캔버스의 키 눌림 이벤트를 처리하는 이벤트 함수 안에서 충돌 검출 로직도 함께 구현한다. 키가 눌려서 위치가 바뀔 때마다 충돌이 벌어지는지 검사해야 하기 때문이다.

38.4 두 모양 사이의 충돌 검출하기

코드 38-5는 두 모양의 충돌을 검출하는 코드다. 이 코드를 기존 키 처리 부분 뒤에 덧붙여야 한다.

- tkinter 설계상 캔버스에 추가된 모양에는 첫 번째 모양은 1, 두 번째 모양은 2와 같이 모두고유 ID가 부여된다. 캔버스에 처음 추가한 모양은 노란색의 직사각형 모양이다.
- 이 코드의 아이디어는 캔버스의 bbox를 호출하면 캔버스에 추가된 객체를 둘러싸는 바운딩 박스(bounding box)의 좌표를 얻을 수 있다는 것이다.

그 후, 캔버스의 find_overlapping 메서드를 호출하면서 방금 찾은 바운딩 박스 좌표를 인자로 넘긴다.

- find_overlapping 메서드는 인자로 받은 박스와 조금이라도 겹치는 모든 모양의 ID들을 돌려준다. 이 코드에서는 박스가 모양 중 하나의 좌표와 같기 때문에, find_overlapping을 실행하면 모양이 겹치는지 여부에 따라 (1,)이나 (1,2)와 같은 튜플이 나올 것이다.

38.4 두 모양 사이의 충돌 검출하기


```
def handle_key(event):
    # 코드 38-3의 기존 키 이벤트 처리 로직 부분 생략
    yellow_xy = canvas.bbox(1) ---- 모양 중 하나를 둘러싼 바운딩 박스의 좌표 구함
    overlapping = canvas.find_overlapping(                    바운딩 박스와 겹치는 모양의 ID를 모두 찾기
                  yellow_xy[0],yellow_xy[1],yellow_xy[2],yellow_xy[3]) -----
    if 2 in overlapping: ---- 겹치는 ID들 중에 다른 사용자가 조종하는 모양의 ID가 들어 있는지 검사
        canvas.create_text(100,100,font=("Arial",20),text="태그!") ---- 캔버스에 텍스트 더하기
```

코드 38-5
충돌 검출하기

38.4 두 모양 사이의 충돌 검출하기

» 한 모양이 다른 모양에 닿자마자 화면이 그림 38-3처럼 변할 것이다.

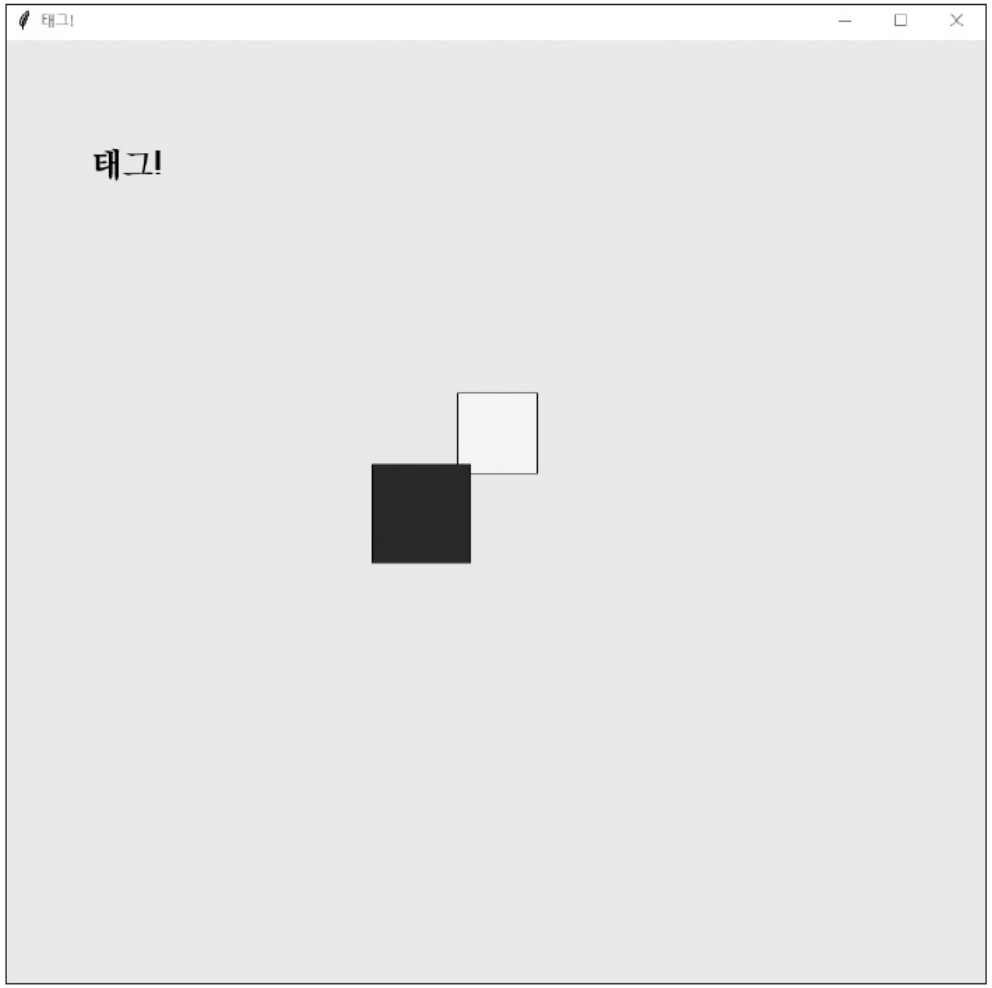

그림 38-3
한 모양의 바운딩 박스와 다른 모양이 서로 겹치면 '태그!'라는 텍스트가 캔버스에 표시된다

38.5 개선할 수 있는 부분

38.5 개선할 수 있는 부분

» 이 게임을 개선할 수 있는 여지는 많다. 이 장에서 코딩한 내용에서 시작하면 좋을 것이다. 몇 가지 아이디어를 제시한다.

- 게임을 다시 하려면 창을 닫고 프로그램을 다시 실행해야 한다. 대신 버튼을 추가해서 사용자가 게임을 다시 할지 결정하게 만들 수 있다. 사용자가 버튼을 클릭하면 두 플레이어의 크기와 위치를 임의로 다시 변경한 후 게임을 시작한다.
- 충돌이 일어났더라도 도망갈 기회를 준다. 두 모양이 서로 충돌한 직후 다시 서로 떨어지면 화면에 표시했던 '태그!' 텍스트를 캔버스에서 없앤다.
- 사용자가 자신의 말의 색을 바꾸거나 모양을 원으로 선택할 수 있게 한다.

38.6 요약

38.6 요약

- 캔버스를 사용해 GUI에모양을 더했고, 이벤트 핸들러를 사용해 사용자가 누른 키에 따라 모양을 이리저리 이동할 수 있다.
- 캔버스에서 여러 모양 사이의 충돌을 어떻게 검출하는지 살펴봤다.
- 코드에서 재활용할 것으로 예상되는 주요 부분에 대해 클래스와 함수를 활용함으로써, 깔끔하고 더 조직적이며 읽기 쉬운 코드를 만드는 방법을 살펴봤다.