```python
# 1. 모든 공을 없애면 게임 종료
# 2. 캐릭터가 공에 닿으면 게임 종료
# 3. 시간 제한 99 초 초과 시 게임 종료


import pygame
import os
####################################################################
# 기본 초기화 (반드시 해야 하는 것들)

pygame.init()

# 화면 크기 설정
screen_width = 640
screen_height = 480
screen = pygame.display.set_mode((screen_width, screen_height))

# 화면 타이틀 설정
pygame.display.set_caption("Nado Pang") # 게임 이름


clock = pygame.time.Clock()
####################################################################

# 1. 사용자 게임 초기화 (배경 화면, 게임 이미지, 좌표, 속도, 폰트 등)
current_path = os.path.dirname(__file__) # 현재 파일의 위치 반환
image_path = os.path.join(current_path, "images") # 이미지 폴더 위치 반환

# 배경 만들기
background = pygame.image.load(os.path.join(image_path, "background.png"))

# 스테이지 만들기
stage = pygame.image.load(os.path.join(image_path, "stage.png"))
stage_size = stage.get_rect().size
stage_height = stage_size[1] # 스테이지 높이 위에 캐릭터를 두기 위해 만들어서 사용

# 캐릭터 만들기
character = pygame.image.load(os.path.join(image_path, "character.png"))
character_size = character.get_rect().size
character_width = character_size[0]
character_height = character_size[1]
character_x_pos = (screen_width / 2) - (character_width / 2)
character_y_pos = screen_height - character_height - stage_height

# 캐릭터 이동 방향 (좌우로만)
character_to_x = 0

# 캐릭터 이동 속도
character_speed = 5

# 무기 만들기
weapon = pygame.image.load(os.path.join(image_path, "weapon.png"))
weapon_size = weapon.get_rect().size
weapon_width = weapon_size[0] #

# 무기는 한 번에 여러 발 발사 가능
weapons = [] # 무기가 여러개 있기 때문에 list 로 정의해줘야한다
```

```python
# 무기 이동 속도
weapon_speed = 10

# 공 만들기 (4 개 크기에 대해 따로 처리)
ball_images = [
    pygame.image.load(os.path.join(image_path, "balloon1.png")),
    pygame.image.load(os.path.join(image_path, "balloon2.png")),
    pygame.image.load(os.path.join(image_path, "balloon3.png")),
    pygame.image.load(os.path.join(image_path, "balloon4.png"))

]

# 공 크기에 따른 최초 스피드 / 좌우로는 같은 속도 , 위 아래만 다름
ball_speed_y = [-18, -15, -12, -9] # index 0, 1, 2, 3 에 해당하는 값

# 공들
balls = []

# 최초 발생하는 큰 공 추가
balls.append({
    "pos_x" : 50,     # 공의 x 좌표
    "pos_y" : 50,     # 공의 y 자표
    "img_idx" : 0,    # 몇번째 이미지를 쓸지 / 공의 이미지 인덱스
    "to_x" : 3,       # 공의 x 축 이동방향, -3 이면 왼쪽 3 이면 오른쪽
    "to_y" : -6,      # y 축 이동방향
    "init_spe_y" : ball_speed_y[0]   # y 의 최초 속도
})

# 사라질 무기, 공 정보 저장 변수
weapon_to_remove = -1
ball_to_remove = -1

# Font 정의
game_font = pygame.font.Font(None, 40)
total_time = 100
start_ticks = pygame.time.get_ticks() # 시작 시간 정의

# 게임 종료 메세지
# TimeOut(시간 초과) / Mission Complete(성공) / Game Over(게임 실패)
game_result = "Game Over"

running = True
while running :
    dt = clock.tick(30)

    # 2. 이벤트 처리 (키보드, 마우스 등)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT: # 캐릭터를 왼쪽으로
                character_to_x -= character_speed
            elif event.key == pygame.K_RIGHT:
                character_to_x += character_speed
            elif event.key == pygame.K_SPACE: # 무기 발사
```

```
            weapon_x_pos = character_x_pos + (character_width / 2) - (weapon_width / 2)
            weapon_y_pos = character_y_pos
            weapons.append([weapon_x_pos, weapon_y_pos])

    if event.type == pygame.KEYUP:
        if event.key == pygame.K_LEFT or event.key == pygame.K_RIGHT:
            character_to_x = 0




# 3. 게임 캐릭터 위치 정의
character_x_pos += character_to_x

if character_x_pos < 0 :
    character_x_pos = 0
elif character_x_pos > screen_width - character_width:
    character_x_pos = screen_width - character_width

# 무기 위치 조정
# 100, 200 -> x 좌표는 그대로 100 , y 좌표는 speed 만큼 빼준다는( 180, 160, 140 ......)
weapons = [[w[0], w[1] - weapon_speed] for w in weapons ] # 무기 위치를 위로

# 천장에 닿은 무기 없애기
weapons = [[w[0], w[1]] for w in weapons if w[1] > 0]

# 공 위치 정의
for ball_idx, ball_val in enumerate(balls) :
    ball_pos_x = ball_val["pos_x"]
    ball_pos_y = ball_val["pos_y"]
    ball_img_idx = ball_val["img_idx"]

    ball_size = ball_images[ball_img_idx].get_rect().size
    ball_width = ball_size[0]
    ball_height = ball_size[1]

    # 가로벽에 닿았을 때 공 이동 위치 변경 (튕겨 나오는 효과)
    if ball_pos_x < 0 or ball_pos_x >screen_width - ball_width:
        ball_val["to_x"] = ball_val["to_x"] * -1

    # 세로 위치
    # 스테이지에 튕겨서 올라가는 처리
    if ball_pos_y >= screen_height - stage_height - ball_height:
        ball_val["to_y"] = ball_val["init_spe_y"]
    else: # 그 외의 모든 경우에는 속도를 증가
        ball_val["to_y"] += 0.5

    ball_val["pos_x"] += ball_val["to_x"]
    ball_val["pos_y"] += ball_val["to_y"]



# 4. 충돌 처리

# 캐릭터 rect 정보 업데이트
character_rect = character.get_rect()
character_rect.left = character_x_pos
```

```
    character_rect.top = character_y_pos

    for ball_idx, ball_val in enumerate(balls) :
        ball_pos_x = ball_val["pos_x"]
        ball_pos_y = ball_val["pos_y"]
        ball_img_idx = ball_val["img_idx"]

        # 공 rect 정보 업데이트
        ball_rect = ball_images[ball_img_idx].get_rect()
        ball_rect.left = ball_pos_x
        ball_rect.top = ball_pos_y

        # 공과 캐릭터 충돌 처리
        if character_rect.colliderect(ball_rect):
            running = False
            break

        # 공과 무기들 충돌 처리
        for weapon_idx, weapon_val in enumerate(weapons):
            weapon_pos_x = weapon_val[0]
            weapon_pos_y = weapon_val[1]

            # 무기 rect 정보 업데이트
            weapon_rect = weapon.get_rect()
            weapon_rect.left = weapon_pos_x
            weapon_rect.top = weapon_pos_y  #// 왜 rect.right 하면 실행했을때 스페이스바 누르면 없어짐?

            # 충돌 체크
            if weapon_rect.colliderect(ball_rect):
                weapon_to_remove = weapon_idx # 해당 무기 없애기 위한 값 설정
                ball_to_remove = ball_idx     # 해당 공 없애기 위한 값 설정

                # 가장 작은 크기의 공이 아니라면 다음 단계의 공으로 나눠주기
                if ball_img_idx < 3 :
                    # 현재 공 크기 정보를 가지고 옴
                    ball_width = ball_rect.size[0]
                    ball_height = ball_rect.size[1]

                    # 나눠진 공 정보
                    small_ball_rect = ball_images[ball_img_idx + 1].get_rect()
                    small_ball_width = small_ball_rect.size[0]
                    small_ball_height = small_ball_rect.size[1]

                    # 왼쪽으로 튕겨나가는 작은 공
                    balls.append({
                        "pos_x" : ball_pos_x + (ball_width / 2) - (small_ball_width / 2),     # 공의 x
좌표
                        "pos_y" : ball_pos_y + (ball_height / 2) - (small_ball_height / 2),     # 공의 y
자표
                        "img_idx" : ball_img_idx + 1,    # 몇번째 이미지를 쓸지 / 공의 이미지 인덱스
                        "to_x" : -3,       # 공의 x 축 이동방향, -3 이면 왼쪽 3 이면 오른쪽
                        "to_y" : -6,      # y 축 이동방향
                        "init_spe_y" : ball_speed_y[ball_img_idx + 1]   # y 의 최초 속도
                    })

                    # 오른쪽으로 튕겨나가는 작은 공
                    balls.append({
```

```
                        "pos_x" : ball_pos_x + (ball_width / 2) - (small_ball_width / 2),     # 공의 x
좌표
                        "pos_y" : ball_pos_y + (ball_height / 2) - (small_ball_height / 2),     # 공의 y
자표
                        "img_idx" : ball_img_idx + 1,    # 몇번째 이미지를 쓸지 / 공의 이미지 인덱스
                        "to_x" : 3,       # 공의 x 축 이동방향, -3 이면 왼쪽 3 이면 오른쪽
                        "to_y" : -6,      # y 축 이동방향
                        "init_spe_y" : ball_speed_y[ball_img_idx + 1]   # y 의 최초 속도
                    })
                break

        # 충돌된 공 or 무기 없애기
        if ball_to_remove > -1 :
            del balls[ball_to_remove]  # 리스트에서 해당 값이 없어진다
            ball_to_remove = -1

        if weapon_to_remove > -1 :
            del weapons[weapon_to_remove]
            weapon_to_remove = -1

        # 모든 공을 없앤 경우 게임 종료 (성공)
        if len(balls) == 0 :
            game_result = "Mission Complete"
            running = False



    # 5. 화면에 그리기 / 코드 순서대로 그려짐
    screen.blit(background, (0, 0))

    for weapon_x_pos, weapon_y_pos in weapons :
        screen.blit(weapon, (weapon_x_pos, weapon_y_pos))

    for idx, val in enumerate(balls):
        ball_pos_x = val["pos_x"]
        ball_pos_y = val["pos_y"]
        ball_img_idx = val["img_idx"]
        screen.blit(ball_images[ball_img_idx], (ball_pos_x, ball_pos_y))

    screen.blit(stage, (0, screen_height - stage_height))
    screen.blit(character, (character_x_pos, character_y_pos))


    # 경과 시간 계산
    elapsed_time = (pygame.time.get_ticks() - start_ticks) / 1000 # ms -> s
    timer = game_font.render("Time : {}".format(int(total_time - elapsed_time)), True, (40, 40, 40))
    screen.blit(timer, (10, 10))

    # 시간 초과했다면
    if total_time - elapsed_time < 0:
        game_result = "Time Over"
        running = False

    pygame.display.update()

# 게임 오버 메시지
msg = game_font.render(game_result, True, (255, 255, 0)) # 노란색
```

```
msg_rect = msg.get_rect(center = (int(screen_width / 2), int(screen_height / 2)))
screen.blit(msg, msg_rect)
pygame.display.update()

# 2 초 대기
pygame.time.delay(2000)

pygame.quit()
```