





1. 제작 목표

스테이지 내에 플랫폼을 밟으면서 별을 모두 모으는 '바운스볼' 플랫폼 게임 만들기

2. Game Design & Structure

- 1)  플레이어: 빨간색 공으로, 스테이지 내에 있는 모든 별을 먹어야 합니다. 좌 우 방향키로 이동할 수 있으며, 평소에는 떨어지는 데, 땅에 닿으면 위로 튕깁니다.
- 2)  플레이어의 목표: 별은 플레이어가 먹어야하는 목표로, 플레이어가 스테이지 내의 모든 별을 먹으면 다음 스테이지로 넘어갑니다. 플레이어가 별을 다 먹지 못하고 사망하면, 해당 스테이지의 별들이 재생성됩니다.
- 3)  땅은 플랫폼으로, 직사각형이며 플레이어와 부딪치면 다음을 따릅니다
 1. 플레이어가 옆면에 부딪치면 통과하지 못하게 막아야 합니다.
 2. 플레이어가 윗면에 부딪치면 플레이어인 공이 위로 튀어오를 수 있게 해야합니다.
 3. 플레이어가 아랫면에 부딪치면 튀어올라서 부딪히기 전의 속도만큼 반대로 튕깁니다.
- 4)  가시는 플레이어를 방해하는 요소로, 가시에 플레이어가 닿으면 플레이어는 죽습니다.
- 5) 그 외로, 플레이어가 화면 바깥으로 나가면 죽습니다.
- 6) 스테이지는 총 6개로 구성됩니다. 각 스테이지마다의 모든 별을 플레이어가 먹으면 다음 스테이지로 넘어갑니다. 스테이지가 모두 끝난 후에는 플레이어의 게임을 플레이한 시간과 다시 시작할 것인지 게임을 끝낼 것인지 고르라는 글이 나옵니다.

3. Code Description

```
1 import pygame
2 import sys
3 import math
4 from pygame.locals import *
5 # 게임 초기화
6 pygame.init()
7 # 화면 크기 설정
8 screen_width = 800
9 screen_height = 600
10 screen = pygame.display.set_mode((screen_width, screen_height))
11 pygame.display.set_caption('바운스볼 게임')
12
13 # 색 정의
14 WHITE = (255, 255, 255)
15 RED = (255, 0, 0)
16 GREEN = (0, 255, 0)
17 YELLOW = (100, 100, 0)
18 BLACK = (0, 0, 0)
19 # 중력 가속도
20 gravity = 0.6
21 # 시간 설정
22 clock = pygame.time.Clock()
23 FPS = 60 # 시간당 프레임
```

파이 게임 초기 세팅을 합니다.

플레이어 클래스를 정의합니다.

```
25 # 플레이어(공)의 초기 위치와 초기 속도 및 튀어오르는 속도 세팅
26 class player:
27     def __init__(self, x, y):
28         self.died = False # 플레이어가 사망했는지
29         self.pos = [x,y] # 플레이어의 위치
30         self.radius = 10 # 플레이어(공)의 반지름
31         self.velocity = [0, 0] # 초기 속도
32         self.bouncing_velocity = -8 # 튀어오르는 속도
33         self.rect = pygame.Rect(0, 0, self.radius * 2, self.radius * 2) # 플레이어의 사각충돌체
34         self.rect.center = (x,y) # 충돌체 위치 설정
35     def updatecollider(self): # 공의 움직임에 따른 충돌체 위치 설정
36         self.rect.center = (self.pos[0], self.pos[1])
```

땅(플랫폼)의 클래스를 정의합니다.

```
39 class platform:
40     def __init__(self, x, y, width, height):
41         self.pos = [x,y]
42         self.rect = pygame.Rect(0, 0, width, height)
43         self.rect.center = (x, y) # 중심점 기준 좌표로 위치 세팅
44         self.width = width
45         self.height = height
46
```

```
47 def checkcollide(self, ball): # 공과의 충돌 판정하고 플랫폼의 어디에 부딪혔냐에 따른 실행
48     if self.rect.collidepoint(ball.rect.midright): # 플랫폼 왼쪽에 부딪칠때
49         ball.velocity[0] = -0.5
50     elif self.rect.collidepoint(ball.rect.midleft): # 플랫폼 오른쪽에 부딪칠때
51         ball.velocity[0] = 0.5
52     elif self.rect.collidepoint(ball.rect.midtop) or self.rect.collidepoint(ball.rect.topleft) or self.rect.collidepoint(ball.rect.topright): # 플랫폼 아래쪽에 부딪칠때
53         ball.pos[1] = self.rect.bottom + ball.radius
54         if (ball.velocity[1] < 0):
55             ball.velocity[1] = -ball.velocity[1]
56     elif self.rect.collidepoint(ball.rect.midbottom) or self.rect.collidepoint(ball.rect.bottomleft) or self.rect.collidepoint(ball.rect.bottomright): # 플랫폼 위쪽에 부딪칠때
57         ball.velocity[1] = ball.bouncing_velocity
58
```

- 이때, 플랫폼의 꼭짓점과 플레이어의 꼭짓점과의 원활한 판정을 위해서 collidepoint를 사용해, 플레이어의 충돌체인 사각형에서 상단 중앙, 상단 왼쪽 꼭짓점, 상단 오른쪽 꼭짓점이 플랫폼의 아래쪽에 부딪치면 플레이어가 플랫폼을 통과하지 못하도록, 위치를 (플랫폼 하단 + 플레이어의

반지름)으로 해주어, 플랫폼 하단과 플레이어의 상단이 딱 맞춰지도록 했습니다. 또한 부딪칠 때의 속도를 반대방향으로 줘서, 부딪치는 느낌이 나도록 했습니다.

그에 반대로, 플레이어 충돌체의 하단 3가지 점(하단 중앙, 하단 왼쪽 꼭짓점, 하단 오른쪽 꼭짓점)이 플랫폼의 위쪽에 부딪치면, 위로 튀어 오르도록 했고, 양옆에 부딪히면 통과하지 못하도록 했습니다. 따라서, 공의 끝부분과 땅의 끝부분이 닿았을 때 판정이 일어나는 것을 보실 수 있습니다(덕분에 좋은 판정이 후합니다). 양옆 판정은 플레이어 충돌체의 옆면의 중앙점으로 따졌습니다. 그렇기 때문에 상, 하단보다 먼저 따져서 옆면과 옆면이 부딪친 판정도 잘 나오도록 했습니다. 충돌판정을 colliderect으로 하면, 원치 않은 판정이 많이 생겨서 이처럼 구현했습니다.

별의 클래스를 정의합니다.

```
58
59 class star:
60     def __init__(self, x, y):
61         self.pos = [x,y]
62         self.img = pygame.image.load('star.png') # 별 이미지 로드
63         self.rect = self.img.get_rect() # 이미지에 따른 충돌체 설정
64         self.rect.center = (x, y) # 중심점 기준 좌표로 위치 세팅
65         self.touch = False # 플레이어가 먹었다면
66
67     def checkcollide(self, ball):
68         if self.rect.colliderect(ball.rect):
69             self.touch = True
70             self.rect = Rect(0,0,0,0) # 충돌체 초기화
71             return True
72         else:
73             return False
```

- self.touch는 별이 플레이어와 닿으면 True로 변하며, 별을 먹었는데 또 플레이어와 충돌판정이 일어나지 않도록 충돌체를 초기화 해줍니다.

가시의 클래스를 정의합니다.

```
75 class spike: # 가시
76     def __init__(self, x, y):
77         self.pos = [x,y]
78         self.img = pygame.image.load('spike.png')
79         self.rect = self.img.get_rect()
80         self.rect.center = (x, y)
81     def checkcollide(self, ball):
82         if self.rect.colliderect(ball.rect):
83             return True
84         else:
85             return False
```

1 스테이지 초기 세팅(스테이지에 들어갈 플랫폼, 별, 가시를 리스트 안에서 만듭니다. 추후에 그릴 때 for문으로 편하게 다 그리기 위해서)

```
86
87 # 플레이어 생성
88 ballinitx, ballinity = 300, 250
89 ball = player(ballinitx, ballinity)
90 # 플랫폼 생성
91 # 정사각형의 플랫폼(20,20) 사이 자연스러운 최대 길이차 = 100,
92 # 50 높은 곳은 (이전 플랫폼의 x좌표 + 이전 플랫폼의 너비 - 20 + 80)
93 ground_list = []
94     platform(200,300,20,100), # 왼쪽 벽
95     platform(600,300,20,100), # 우측 벽
96     platform(400,360,420,20), # 바닥
97
98
99     platform(70,60,20,60),
100     platform(110,40,60,20),
101     platform(110,80,60,20),
102     platform(130,100,20,60),
103     platform(90,120,60,20),      # S
104
105     platform(200,80,20,100),      # I
106
107     platform(310,40,90,20), |
108     platform(270,80,20,100),
109     platform(310,80,20,100),
110     platform(350,80,20,100),      # M
111
112     platform(410,80,20,100),
113     platform(440,40,60,20),
114     platform(440,80,60,20),
115     platform(480,60,20,60),      # P
116
117     platform(540,80,20,100),
118     platform(580,120,60,20),      # L
```

```

119     platform(670,80,20,100),
120     platform(710,80,60,20),
121     platform(710,40,60,20),
122     platform(710,120,60,20),    # E
123 ]
124
125 star_list = [
126     star(580,280)
127 ]
128 spike_list = []
129 goals = 0    # 별을 먹은 개수
130 # 다른 스테이지부터 로드하고 싶다면
131 stage = 1 # 스테이지 숫자를 바꾸고 (초기 1)
132 loadstage = False # 로드 스테이지를 True로 하면 된다. (초기 False)
133 stageclear = False # 스테이지를 클리어했는지?
134 getendtime = False # 스테이지를 클리어할때 시간을 구했는지?(다음 스테이지로 가기전 딜레이를 위한 것)
135 gameover = False
136 playtime = 0

```

게임 루프 시작

```

137 # 게임 루프
138 while True:
139     clock.tick(FPS) # 프레임 제한
140     # 공이 죽으면
141     if (ball.died):
142         if(abs(pygame.time.get_ticks() - diedtime) > 800): # 0.8초가 지나면 리스폰(get_ticks는 밀리초를 반환 1초는 1000밀리초)
143             ball.pos = [ballinitx, ballinity]
144             ball.updatecollider()
145             ball.died = False
146             goals = 0
147             print('초기화')
148             for staridx in star_list:
149                 if staridx.touch: # 죽기 전에 먹은 별들 다시 초기화
150                     staridx.touch = False
151                     staridx.rect = Rect(0,0,20,20)
152             staridx.rect.center = staridx.pos[0], staridx.pos[1]
153

```

- 공(플레이어)이 죽으면 0.8초뒤에 해당 스테이지의 초기 위치에서 리스폰합니다. 먹었던 별들도 다시 생겨나야 합니다.

```

155     # 스테이지 클리어시 다음 스테이지로 가기전 약간의 딜레이를 위한 endtime 얻기
156     if (stageclear and not getendtime):
157         endtime = pygame.time.get_ticks()
158         getendtime = True # endtime을 얻었는가?
159         print('종료중')
160     # 딜레이가 끝나고 나면 다음 스테이지 로드
161     if (getendtime and stageclear):
162         if(abs(pygame.time.get_ticks() - endtime) > 800):
163             stage += 1
164             print(stage)
165             loadstage = True
166             stageclear = False
167             getendtime = False

```

- stageclear는 별을 다 먹으면 True가 됩니다.

- 0.8초가 지나면 스테이지를 1단계 높이고, 스테이지를 로드하겠다는 의미의 loadstage를 True로 합니다.

다음 스테이지로 초기화합니다.

```
168         if(stage == 2 and loadstage):    # 스테이지 2 맵으로 세팅
169             ground_list.clear()
170             star_list.clear()
171             spike_list.clear()
172             goals = 0
173             ballinitx, ballinity = 110, 450
174             ball.pos = [ballinitx, ballinity]
175             ball.updatecollider()
176             ball.velocity = [0, 0]
```

- 다음 스테이지로 초기화하기 위해서 땅리스트와 별리스트, 가시리스트, 별을 먹은 개수를 비워준다. 그리고 다음 스테이지의 공의 초기 위치를 세팅해줍니다.

```
177     ground_list = []
178         platform(70,460,20,100), # 왼쪽 벽
179         platform(100,500,80,20), # 첫 바닥
180         platform(200,500,60,20),
181         platform(300,500,60,20),
182         platform(420,500,60,20),
183         platform(550,500,60,20),
184         platform(710,500,60,20),
185         platform(740,460,20,100),
186
187         platform(100,80,20,100),
188         platform(130,80,40,20),
189         platform(160,80,20,100), # H
190
191         platform(220,80,20,100),
192         platform(260,80,60,20),
193         platform(260,40,60,20),
194         platform(260,120,60,20), # E
195
196         platform(340,80,20,100),
197         platform(380,120,60,20), # L
198
199         platform(460,80,20,100),
200         platform(500,120,60,20), # L
201
202         platform(580,80,20,100),
203         platform(620,120,80,20),
204         platform(650,80,20,100),
205         platform(620,40,80,20), # O
206
207         platform(700, 60,20,60),
208         platform(700, 120, 20, 20) # !
209     ]
210     star_list = []
```

```
210     star_list = [
211         star(300,420),
212         star(420,420),
213         star(550,420),
214         star(720,420)
215     ]
```

- 그리고 난 후에 넣고 싶은 땅, 별, 가시를 세팅해줍니다. (2스테이지에는 가시를 넣지않아 생략합니다.)

```
216     loadstage = False
```

다 세팅한 후에는 loadstage를 False로 하여 게임을 동작하게 합니다.

다른 스테이지들의 구성도 이와 같습니다.

따라서 생략하겠습니다.

모든 스테이지가 끝나면 플레이시간과 다시 할 것인지 묻는 화면을 제시합니다.

```
445 if(stage == 7 and loadstage): # 스테이지 종료
446     playtime = pygame.time.get_ticks() - playtime # playtime을 재시작할 때 다시 설정해주어야함.
447     gameover = True
448     while(gameover):
449         screen.fill(BLACK)
450         showtime = pygame.font.SysFont('comicans', 30).render("Your PlayTime: " + str(playtime / 1000)+"sec", True, WHITE)
451         showtimerect = showtime.get_rect()
452         showtimerect.center = (400, 250)
453         selectgame = pygame.font.SysFont('comicans', 20).render("Press R key to restart game or Press Q key to quit game", True, WHITE)
454         selectgamerect = selectgame.get_rect()
455         selectgamerect.center = (400, 300)
456         screen.blit(showtime, showtimerect)
457         screen.blit(selectgame, selectgamerect)
458         pygame.display.update()
```

```
459 for event in pygame.event.get():
460     if event.type == pygame.QUIT:
461         pygame.quit()
462         sys.exit()
463     if event.type == pygame.KEYDOWN:
464         if event.key == pygame.K_q: # Q키
465             pygame.quit()
466             sys.exit()
467         if event.key == pygame.K_r: # 재시작
468             ground_list.clear()
469             star_list.clear()
470             spike_list.clear()
471             goals = 0
472             ballinitx, ballinity = 300, 250
473             ball.pos = [ballinitx, ballinity]
474             ball.updatecollider()
475             ball.velocity = [0, 0]
476             ground_list = [ # 스테이지 1로 초기화한다.
477                 platform(200,300,20,100), # 왼쪽 벽
478                 platform(600,300,20,100), # 우측 벽
479                 platform(400,360,420,20), # 바닥
```

이 화면에서 키보드 Q를 누르거나, 종료버튼을 누르면 프로그램을 종료합니다.
R을 누르면 1스테이지부터 다시 시작합니다.
477줄부터는 스테이지 세팅입니다.

```
513     goals = 0
514     stage = 1
515     loadstage = False
516     stageclear = False
517     getendtime = False
518     playtime = pygame.time.get_ticks()
519     gameover = False
```

다른 스테이지 세팅과 다르게 1스테이지부터 다시 하는 것은 게임 루프인 while문 들어오기 전 초기화 세팅도 함께 해줍니다. Playtime은 재시작할 때부터 재야 하므로 재시작을 했을 때의 시간을 미리 재 둡니다.

게임 플레이 도중 종료버튼을 누르면 프로그램을 종료합니다

```
521     for event in pygame.event.get():
522         if event.type == pygame.QUIT:
523             pygame.quit()
524             sys.exit()
525
```

```
525     keys = pygame.key.get_pressed() # 동시에 두 개의 키를 누르고, 다른 키를 했을때 공이 멈추지 않도록
526     if keys[pygame.K_LEFT]: # 하기 위해서 pressed를 사용
527         ball.velocity[0] = -4
528     if keys[pygame.K_RIGHT]:
529         ball.velocity[0] = 4
530     elif not keys[pygame.K_LEFT] and not keys[pygame.K_RIGHT]:
531         ball.velocity[0] = 0
532     # if event.type == pygame.KEYDOWN: # 플레이어 키 입력
533     #     if event.key == pygame.K_LEFT:
534     #         ball.velocity[0] = -5
535     #     elif event.key == pygame.K_RIGHT:
536     #         ball.velocity[0] = 5
537     # elif event.type == pygame.KEYUP and ball.velocity[0] < 0:
538     #     if event.key == pygame.K_LEFT:
539     #         ball.velocity[0] = 0
540     # elif event.type == pygame.KEYUP and ball.velocity[0] > 0:
541     #     if event.key == pygame.K_RIGHT:
542     #         ball.velocity[0] = 0
543     #
```

- 공을 움직이는 조작키 설정입니다. 동시 키를 눌렀을 때의 편한 조작감을 위해서 keydown & up 이 아닌 pressed를 사용했습니다.

```
545 # 공에 중력 가하기(중력에 의해 가해진 속도 최대치 한정하기)
546 # 속도가 너무 커지면 위치가 큰 값으로 순간이동하기 때문에 콜리전 측정 등이 이상해짐.
547 # 8로 정한 이유는 플랫폼들의 두께가 보통 20이기 때문에 2프레임이 지나도 16의 위치가 변하는 것이므로
548 # 충돌 판정이 스킵되지 않을 것이라고 생각했다.
549 if (ball.velocity[1] < 8):
550     ball.velocity[1] += gravity
```

공이 중력에 의한 속도를 제한해두지 않으면 플랫폼을 뚫고 지나가는 현상이 발생함으로 최대 속도를 제한해 둡니다.

게임 속 요소들과 공의 콜리전 판정

```
554 # 플랫폼과 플레이어(공)와의 콜리전 판정
555 for groundidx in ground_list:
556     groundidx.checkcollide(ball)
557 # 별과 플레이어와의 콜리전 판정
558 for staridx in star_list:
559     if staridx.checkcollide(ball):
560         goals += 1
561         print('달음')
562         if(goals == len(star_list)): # 모든 별을 먹으면 stageclear
563             stageclear = True
564 # 가시와 플레이어와의 콜리전 판정
565 for spikeidx in spike_list:
566     if spikeidx.checkcollide(ball) and not ball.died:
567         diedtime = pygame.time.get_ticks() # 리스폰까지의 딜레이를 위한 시간 측정
568         ball.died = True
569         print('가시 달음')
```

공 움직이기

```
571 # <주의> 공 움직이는 것은 콜리전 판정 이후에 있어야함!!
572 if not ball.died:
573     # 공 움직이기
574     ball.pos[0] += ball.velocity[0]
575     ball.pos[1] += ball.velocity[1]
576     # 물체의 움직임에 따른 콜라이더 업데이트
577     ball.updatecollider()
```

- 공을 움직이는 코드가 콜리전 판정 이전에 있으면 콜리전 판정에 따른 움직임이 제가 의도했던 것만 달라져서 판정 이후에 넣었습니다.

화면 아래 바깥을 나가면 죽습니다.

```
578
579 # 화면 가장자리와 공의 콜리전
580 # 플레이어가 화면 바깥으로 나가면 죽음
581 if (ball.pos[1] + ball.radius >= 600) and not ball.died:
582     diedtime = pygame.time.get_ticks() # 리스폰까지의 딜레이를 위한 시간 측정
583     ball.died = True
```

- 그 외의 가장자리는 스테이지마다 갈 수 없거나 가도 금방 돌아와야 하거나, 결국에는 아래 바

깎을 벗어나, 죽도록 했습니다.

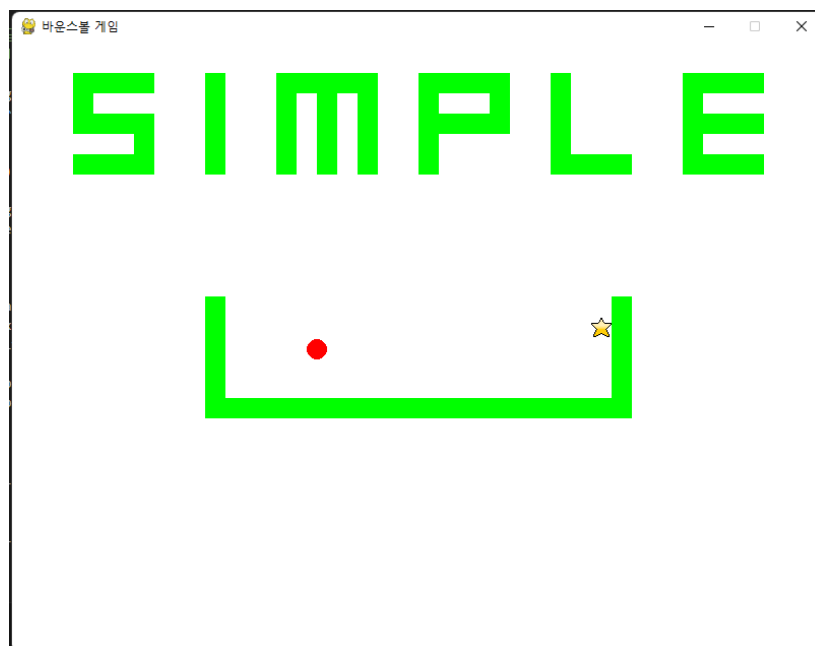
창에 그리기

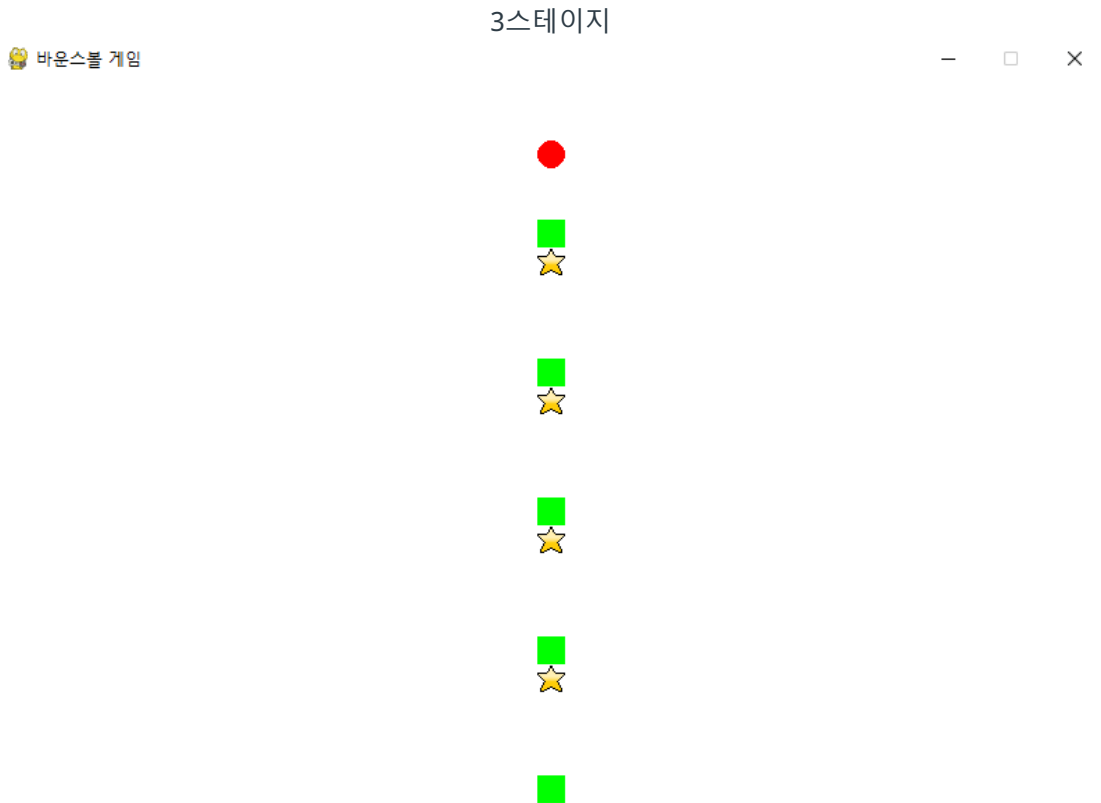
```
585     # 화면 클리어
586     screen.fill(WHITE)
587     # 플랫폼 그리기
588     for groundidx in ground_list:
589         pygame.draw.rect(screen, GREEN, groundidx.rect)
590
591     # 별 그리기
592
593     for staridx in star_list:
594         if not staridx.touch:
595             screen.blit(staridx.img, staridx.rect)
596     # 가시 그리기
597     for spikeidx in spike_list:
598         screen.blit(spikeidx.img, spikeidx.rect)
599
600     # 공그리기
601     if not ball.died:
602         pygame.draw.circle(screen, RED, (int(ball.pos[0]), int(ball.pos[1])), ball.radius)
603
604     # 화면 업데이트하기
605     pygame.display.flip()
606
```

- 별은 touch가 True인 즉, 플레이어가 먹었다면, 더 이상 화면에 보이지 않아야 합니다.
- 공은 죽었다면 더 이상 화면에 보이지 않아야 합니다.

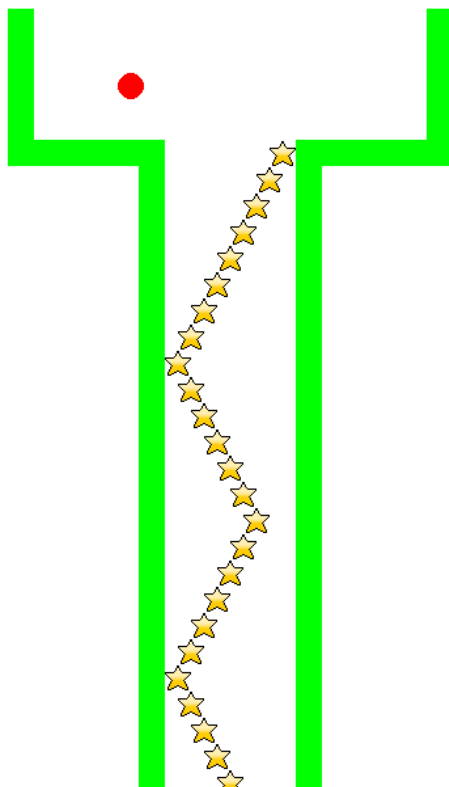
4. Execution Environment

1 스테이지

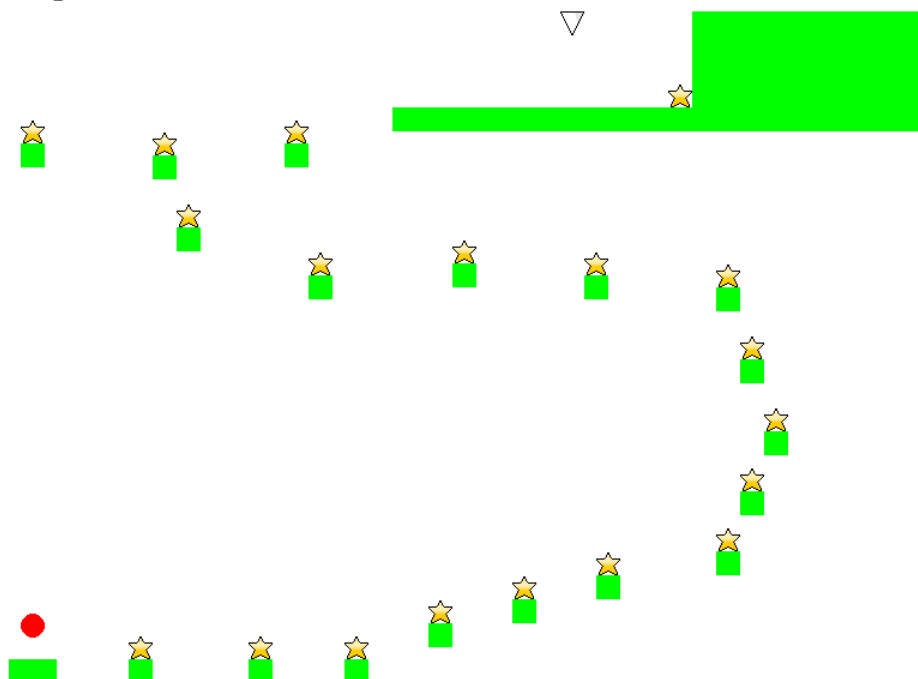


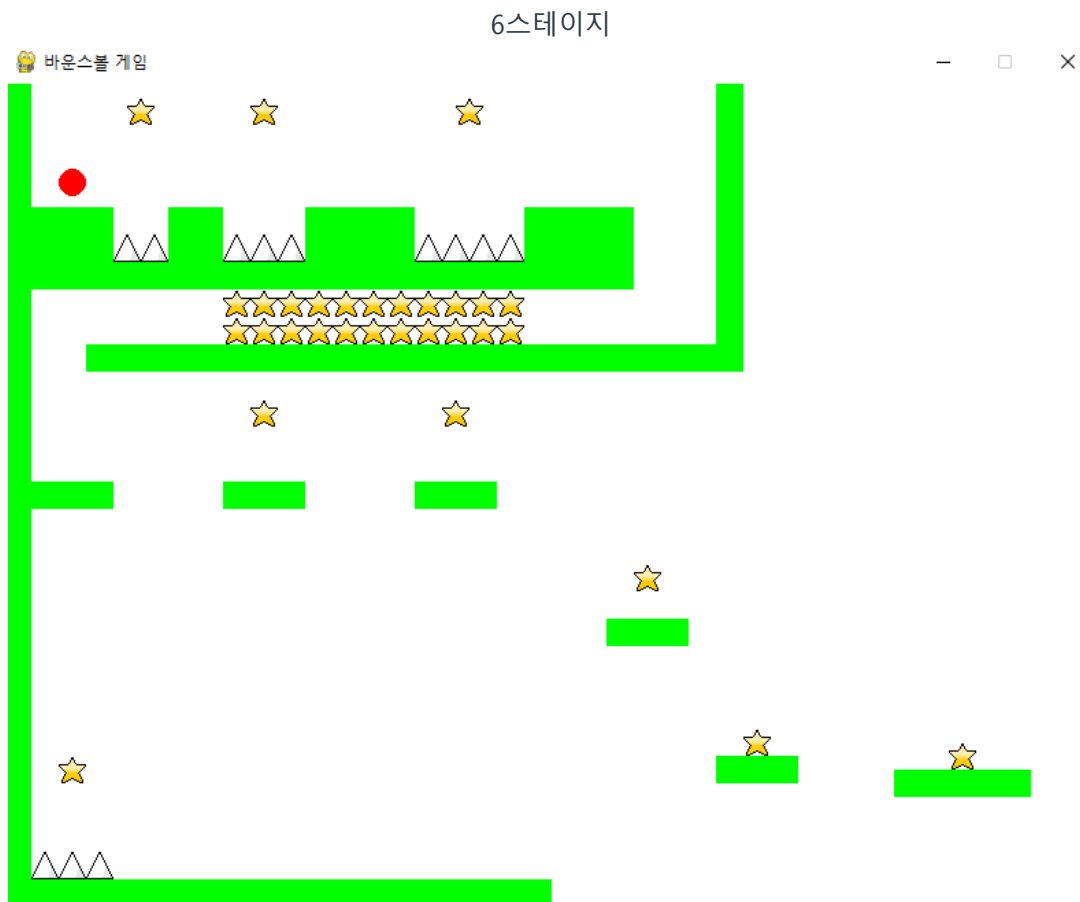


4 스테이지



5스테이지





모든 스테이지 클리어 시

