**결과보고서**

2조

박혁규, 최준우, 임신우, 김수미, 홍진성

< 1. 프로그램 구성요소 >

1) 무궁화 꽃이 피었습니다.

2) 줄다리기

3) 홀짝게임

< 2. 게임 설명 >

1) 무궁화 꽃이 피었습니다.

방향키를 이용하여 캐릭터를 움직일 수 있다.

장애물이 캐릭터의 진로를 방해한다.

신호등(빨간불/초록불)과 음성(on/off)을 이용해 움직일 수 있는 시간 제어

==> 초록불, 음성이 재생될 때, 플레이어가 움직일 수 있다.

==> 잘못된 시간에 움직이면 GAME OVER

클리어 시간에 따른 보상을 구슬로 차등 지급한다.

2) 줄다리기

화면 상단에 입력할 때마다 바뀌는 방향키 이미지에 맞춰 방향키를 입력하면 줄이 player 방향(오른쪽)으로 당겨진다.

일정 시간마다 반대쪽으로 줄이 당겨진다.

방향키를 잘못 입력한 경우에도 줄이 반대쪽으로 당겨진다.

클리어 시간에 따른 보상을 구슬로 차등 지급한다.

3) 홀짝 게임

앞서 1단계와 2단계에서 보상으로 지급받은 구슬을 이용하여 홀짝 게임을 진행한다.

홀짝을 맞출 시 컴퓨터가 숨겼던 구슬의 개수만큼 player가 지급 받는다. 반대로 맞추지 못할 시, 구슬의 개수만큼 player의 구슬이 차감된다.

홀짝 게임을 통해 목표개수 이상의 구슬을 획득하면 게임에서 승리한다.

Player가 소유한 구슬의 개수가 0개가 되면 게임에서 패배한다.

< 3. 소스 설명 >

텍스트이(가) 표시된 사진

자동 생성된 설명

게임 실행에 필요한 모듈인 pygame, time, random 을 import 했다.

텍스트, 실외, 잔디, 스크린샷이(가) 표시된 사진

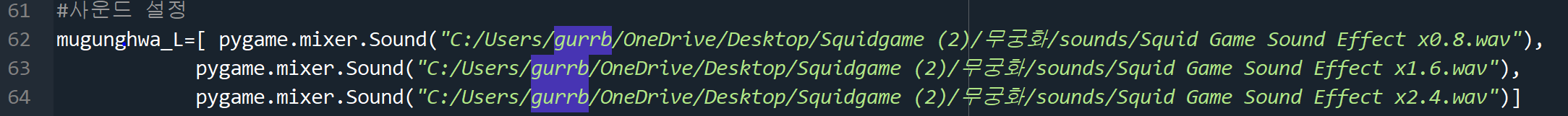
자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

각 단계에 필요한 이미지를 pygame.image.load() 함수를 이용하여 이미지를 불러왔다.

1. 무궁화 꽃이 피었습니다.



게임에 필요한 음성파일을 0.8배, 1.6배, 2.4배 배속 시켜 무작위로 불러오기 위해 리스트 변수 안에 불러왔다.

텍스트이(가) 표시된 사진

자동 생성된 설명

캐릭터의 움직임을 나타내기 위해 움직이는 동작을 나눠 리스트에 저장

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

게임에 나오는 캐릭터, 장애물, 신호등의 크기와 초기 위치를 설정

Ch\_x\_vector, ch\_y\_vector 변수는 캐릭터의 움직임에 대한 정보를 저장할 변수

텍스트이(가) 표시된 사진

자동 생성된 설명

게임에 필요한 변수들을 초기화

텍스트, 실내, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Running이 1 or 0 이냐에 따라 게임 실행여부가 결정

Dt=clock.tick(60)로 프레임을 60fps로 설정하여 게임의 속도를 제어

Stage의 값에 따라 해당 단계의 게임이 실행

if(e\_time>init\_time+term) 현재 시간(e\_time)이 (초록불의 점등시간+초록불 지속 시간) 보다 커지면 조건문을 만족하여 다음 초록불이 켜지는 시간을 초기화 해주며 다음 음성파일 배속을 랜덤으로 결정

350번째 줄에서 현재시간이 초록불이 켜지는 시간에 대한 조건을 만족하면 초록불, 소요시간을 screen.blit()함수를 이용해 화면에 표시하고 음성파일을 실행

텍스트이(가) 표시된 사진

자동 생성된 설명

다음 for문은 Key event에 대한 반복문으로 pygame에서 어떤 이벤트가 발생하면 해당 반복문을 돌게 된다. 각 조건문에 left, right, stop, up, down은 현재 캐릭터가 어느 방향으로 가고 있는지에 대한 정보를 담는다. 또한 lastmotion은 이후 캐릭터가 멈춰 있을 때, 어느 방향을 보고 있을지를 표현해줄 변수이다. ch\_x\_vector, ch\_y\_vector는 임의의 방향키를 눌렀을 때, 설정한 속도만큼 계속 누적하여 저장해준다. 방향키를 떼었을 때는 해당 변수를 0으로 초기화하여 캐릭터의 좌표가 변경되지 않게 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명좌측의 코드를 통해 캐릭터의 좌표가 변경되며

Clock.tick(60)을 통해 반환된 값인 dt를 곱하여 60fps에 맞는 속도로 캐릭터가 이동

텍스트이(가) 표시된 사진

자동 생성된 설명

캐릭터의 위치 좌표가 장애물의 영역 안으로 들어오는 것을 제한해야 하므로 다음 조건문을 작성하여 각각의 장애물로 캐릭터가 들어가는 것을 통제

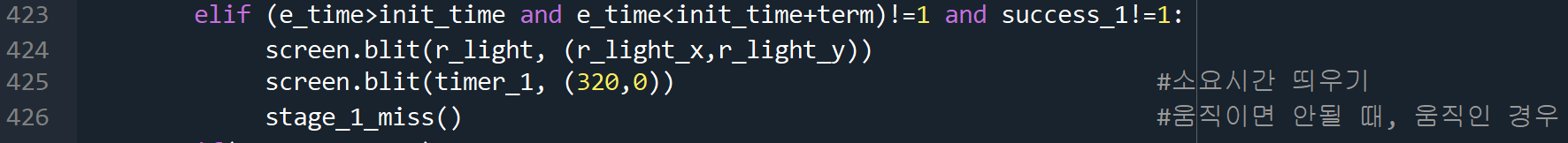
조건문 안에 있는 smoothmove()함수와 limitch() 함수는 아래 사진의 코드를 보면

텍스트이(가) 표시된 사진

자동 생성된 설명

404번 라인에 smoothmove()함수 대신 단순히 ch\_x\_vector=0, ch\_y\_vector=0으로 장애물에 부딪혔을 때, 캐릭터의 움직임을 제한하게 되면 캐릭터가 장애물에 붙었을 때 잘 움직이지 않는 문제가 발생하여 smoothmove함수를 통해 이를 조금 보정

limitch() 함수는 캐릭터가 화면밖으로 넘어가는 것을 막기위한 함수



빨간불일 때에 대한 조건문, 해당 조건을 만족하면 빨간 불이 켜진 신호등, 소요시간이 화면에 띄워진다. Stage\_1\_miss()함수는 빨간 불 일 때의 캐릭터 움직임을 감지

텍스트이(가) 표시된 사진

자동 생성된 설명

빨간 불 일 때의 조건문 내부에서 어떠한 key event라도 입력되면 success\_1변수에 -1값을 할당

또한 키를 눌렀을 때 의 시점(실패한 시점)을 s\_time\_1에 저장하고 success\_1변수에 -2값을 할당주어 다시 s\_time\_1변수가 초기화되는 것을 방지한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

실패시점 s\_time\_1으로부터 3초가 지나기 전까지 fail이모티콘을 띄워주어 플레이어가 게임에서 실패했음을 인지하게 해주고 running에 0을 할당하여 while루프를 나옴

텍스트, 실내, 스크린샷이(가) 표시된 사진

자동 생성된 설명

위 조건문은 캐릭터의 위치 좌표 중, x좌표가 70보다 작을 때, 즉 결승선을 통과했을 때 실행된다.

처음 success\_1변수는 0이 할당되어 있으므로 434라인의 조건문에 들어가 success\_1 변수에 1을 할당해주고 성공시점 s\_time\_1을 초기화 한다. 위의 실패했을 때와 마찬가지로 3초동안 success 이모티콘을 player에게 보여준 후 stage2변수에 2를 할당하여 다음 루프에서 2단계가 실행될 수 있게 한 후, 클리어 시간에 따른 보상을 차등 지급한다. 그리고 2단계 소스에서 필요한 2단계 시작시간 beadtime\_2를 초기화 해놓는다.

텍스트이(가) 표시된 사진

자동 생성된 설명

ch\_Move함수는 위의 key event반복문에서 할당받았던 left,right,up,down,stop변수를 받아와 해당 방향으로의 움직임을 표현해준다.

초기 캐릭터는 결승선 방향인 좌측방향을 보며 시작되기 때문에 movecount의 초기값을 -1로 설정해주면 98번 라인 조건문에서 좌측방향을 보는 캐릭터를 그려준다. 이후 어떤 방향키를 눌러 left또는 right가 참이 될 경우, 60fps이므로 움직임에 대한 사진10장에 각각 6프레임씩 할당하여 보여준다.->스프라이트

위, 아래 방향에서는 lastmotion 변수를 통해 캐릭터가 마지막으로 움직였던 방향을 본 채로 위아래 방향으로 움직이게 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

453~458라인은 screen.blit()함수를 이용하여 장애물들을 화면에 띄워준다.

2. 줄다리기

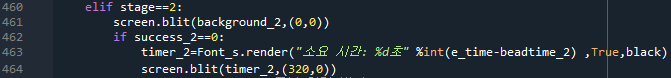
다음은 2단계 줄다리기 게임이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

2단계에 필요한 background, rope사진등이 있으며 ropeL,ropeR은 각각 왼쪽, 오른쪽으로 줄을 당기는 이미지를 따로 저장해 주었다.

Direction 리스트에는 화면에 표시할 화살표를 위, 아래, 오른쪽, 왼쪽 순으로 저장해 두었다.



2스테이지가 실행한 후 background\_2 이미지를 띄워준다.

Success\_2==0이라는 뜻은 게임이 진행중이라는 뜻이며 게임 진행 중 소요시간을 화면 상단에 표시해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

키보드 방향키에 값을 1,2,3,4씩 할당해 준다.

if event.type==pygame.KEYDOWN:

이 조건문은 키보드가 눌렸을 때 파이썬이 입력을 인식하고, 차례대로 K\_UP, K\_DOWN, K\_RIGHT, K\_LEFT는 위쪽 방향키, 아래쪽 방향키, 오른쪽 방향키, 왼쪽 방향키 입력을 감지한다.

순차적으로 1,2,3,4값의 keyboardN을 할당해준다.

반대로 if event.type==pygame.KEYUP: 는 키보드입력이 인식되지 않았을 때 로프의 이동거리를 0으로 제한하고 keyboardN을 0으로 할당시킨다.

텍스트이(가) 표시된 사진

자동 생성된 설명

바깥에 direct=random.randint(1,4)를 통해 처음 방향키의 초기값이 제시가 되고

이 방향키의 값과 키보드 입력의 값이 같다면 로프를 오른쪽으로 당기는 이미지를 표시하고 로프의 x벡터를 +1해준다.

그 후 다시 direct=random.randint(1,4) 를 통해 새로운 방향키 값을 할당한 후 keyboardN의 값을 초기화시킨다.

반대로 실패할 시에는 -0.5만큼 당겨지고 로프를 왼쪽으로 당기는 이미지가 표시된 후 다시 위의 과정을 반복한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

컴퓨터가 시간을 측정해 0.7초마다 -1씩 당긴다.

플레이어가 맞는 방향키를 입력해 x벡터가 500이상이 될 시 success\_2값을 1로 할당하고 승리로 판정,

반대로 x벡터가 300-rope\_w가 될 시 success\_2값을 -1로 할당 후 패배로 판정.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 조건문을 통해 3초동안 성공 이모티콘을 띄운 후 플레이어의 클리어 시간을 판정한다.

이때, 플레이어의 클리어 시간 판정 이전에 3초간 이모티콘을 띄운 시간이 포함되어 있기에 원래의 측정시간에 각각 +3초를 더해준 값을 할당하였다. 클리어 시간에 따라 다음 스테이지에 필요한 보상을 차등지급하였다.

이후 스테이지 3으로 이동한다.

텍스트, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

위 조건문은 플레이어가 실패했을 때 발동된다.

성공과 마찬가지로 실패 이모티콘을 3초간 화면에 띄워주고 3초가 지난 이후 running값을 0으로 만들어 파이썬을 종료시킨다.

그리고 로프의 이동에 dt를 곱해 60fps에 맞는 속도로 로프가 이동.

3. 홀짝 게임

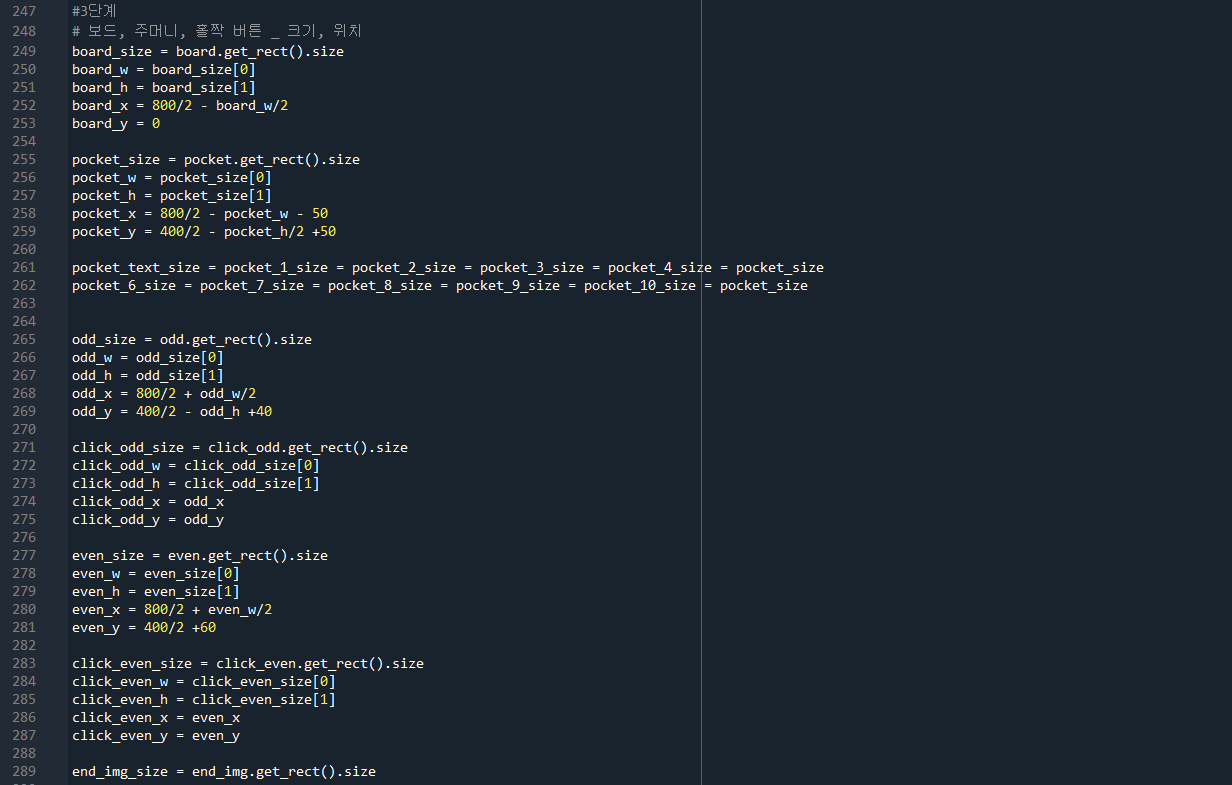
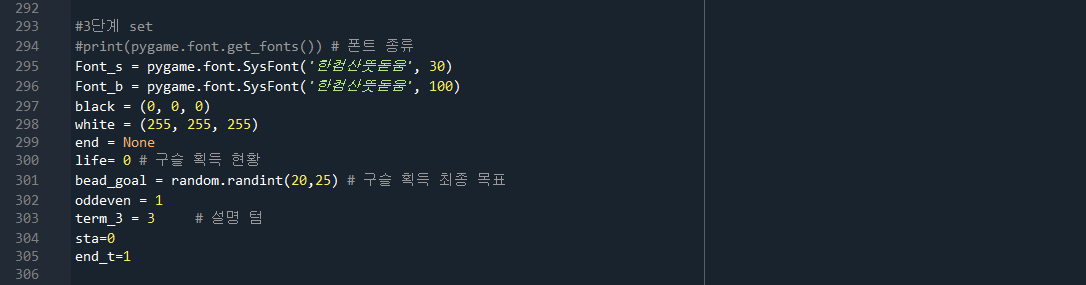
다음은 이 게임의 마지막인 3단계 홀수 짝수 게임이다.

[ while 밖 ]

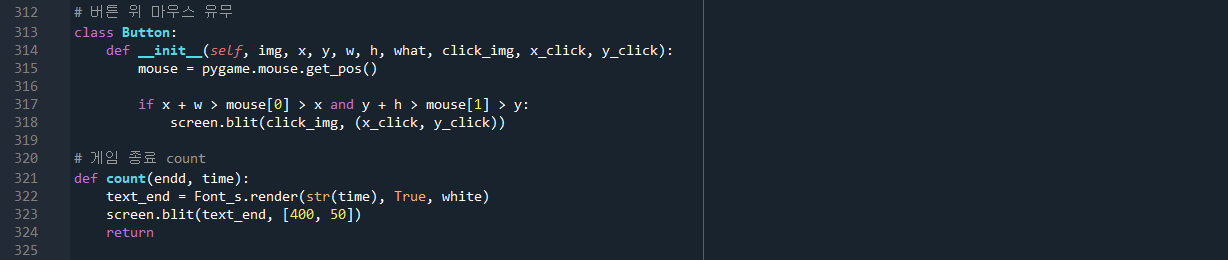
텍스트이(가) 표시된 사진

자동 생성된 설명3단계에서 고정적으로 사용되거나 초기에 필요한 것들을 while문 밖에 설정해 주었다.

(1) 홀짝 게임을 위한 이미지의 크기와 위치 설정



(2) 글자의 크기, 폰트, 색깔과 구슬 획득 현황 / 게임 종료 전후에서 나뉠 ‘end’ / 구슬 획득 현황 ‘life’ / 최종 목표인 20과 25 사이의 구슬 획득 수 ‘bead\_goal’ / 3단계 내에서 게임 진행 상태에 따라 바뀔 ‘oddeven’ / 3단계 시작 초반에 게임 설명을 위해 사용될 시간 텀 ‘term\_3’ / 고정 시간을 만들 때 사용 될 ‘sta’와 ‘end\_t’



(3) 버튼 위의 마우스 유무에 따라 버튼의 이미지를 바꾸어주는 class button을 만들었다. 이는 마우스가 올라오기 전의 이미지와 그 이미지의 x, y, w, h 값을 받은 후 (x)와 (x+w) 사이, (y)와 (y+h)사이에 마우스가 올라온지 확인한다. 확인 후 올라 온 것이 맞다면 screen.blit를 새롭게 지정해준다.

(4) 게임 종료 시 종료까지의 시간을 count하는 count 함수를 만들었다. 이는 시간에 따라 다른 screen.bilt를 새롭게 지정해준다.’

[ while 안 ]

3단계는 게임의 각 진행 상황에 따라 oddeven을 다르게 지정하였다.

oddeven = 1 ) 3단계 내에서 계속 변화할 3과 8사이의 구슬의 수 랜덤으로 지정

oddeven = 2 ) 버튼 클릭 전 후

oddeven = 3 ) 버튼 클릭 후 오답 확인

oddeven = 4 ) 게임 end 여부 확인 – (목표 < 목숨) or (목숨 < 0)시 end

oddeven = 5 ) 홀수 짝수 게임의 결과가 정해진 상태, 1~4 상태를 빠져나가기 위해 존재

텍스트이(가) 표시된 사진

자동 생성된 설명

(0) 3단계 시작 및 홀수 짝수 게임 설명 (564 ~ 507)

564 stsge = 3이 됐으므로 3단계 게임이 시작된다.

565 게임 설명을 위한 배경을 backgound\_3으로 설정한다.

566 while문 안이라서 연속적으로 변화하는 time.time.()값을 고정하기 위해 sta = 0일 때 시간을 측정하고 그직후 sta를 1로 바꾸어준다. init\_time\_hol : 3단계 시작 시간

570 ( time.time() – init\_time\_hol )이 일정 값이 될 때 까지 게임을 설명한다.

(1) oddeven = 1 ( 590 ~ 605 )

590 설명을 위한 일정 시간이 지났으므로 else로 들어오게 된다.

593 oddeven = 1 : 구슬 수 랜덤 지정 후 답 정하기

594 pocket\_state = 0 : 정답을 숨긴 상태의 손 이미지

595 board\_state = 0 : player에게 게임 현황을 알려주는 텍스트

596 bead\_num : 3단계에서 계속 변화할 구슬은 3과 8 사이의 랜덤 수

599 answer : 랜덤으로 지정된 구슬 수에 따라 “홀수” 또는 “짝수”로 답 지정

605 구슬 수와 답을 정하였으므로 oddeven = 2로 바꾸어 준다.



(2) oddeven = 2 ( 608 ~ 632 )

608 oddeven = 2 : 버튼 클릭을 통해 답을 입력 받음

609 pygame.mouse.get\_pos() : 마우스 커서 위치를 얻는다

610 pygame.event.poll() : 단일 이벤트를 얻는다. 버튼이 클릭되었는지 확인하기 위해 존재

616 짝수 버튼의 x축 범위와 y축 범위내에 마우스가 들어왔는지 확인한다.

617 마우스가 범위에 들어왔다면 마우스로 클릭되었는지 확인한다.

618 pocket\_state = None : 구슬이 보이지 않는 손 이미지

619 홀수 버튼을 클릭하면 player의 선택은 “홀수”가 된다.

620 답을 입력 받았으므로 oddeven = 3으로 바꾸어 준다.

622 짝수 버튼도 홀수 버튼과 동일한 방식으로 작동된다.

631 우측 상단의 종료를 누르면 running이 0이 됨으로써 while문의 작동이 멈추고 게임 종료



(3) oddeven = 3 ( 636 ~ 649 )

636 oddeven = 3 : 3단계의 정답과 player 선택 비교

637 답안 입력과 정답 제시의 사이에 1초 간격을 두기 위해 delay(1000) 사용

638 정답에 맞는 구슬 수가 포함된 손 이미지로 변경

640 만약 정답과 player 선택이 동일하다면

642 board\_state = 1 : 칠판은 구슬의 수, 정답 및 “정답” 안내 텍스트로 변경

643 life는 구슬 수만큼 증가

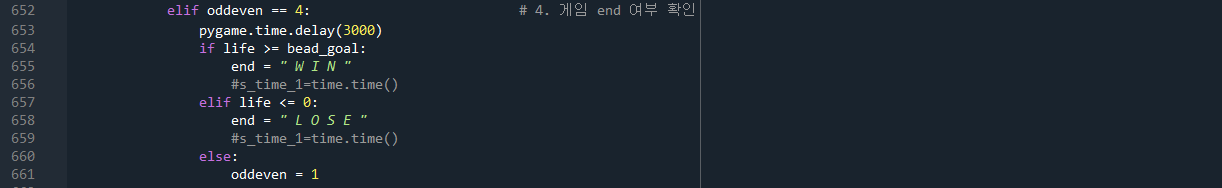
645 만약 정답과 player 선택이 동일하지 않다면

647 board\_state = 2 : 칠판은 구슬의 수, 정답 및 “오답” 안내 텍스트로 변경

648 life는 구슬 수만큼 감소

649 정답과 선택 비교 후 정답 여부 안내가 끝남에 따라 oddeven = 4로 바꾸어 준다.

(4) oddeven = 4 ( 652 ~ 661 )



652 oddeven = 4 : 게임 종료 여부를 확인

653 정답 확인과 게임 종료 여부 확인 사이에 3초 간격을 두기 위해 delay(3000) 사용

654 만약 목숨 값이 목표 값 보다 커진다면 “WIN”이 된다.

657 만약 목숨 값이 0 이하가 된다면 “LOSE”가 된다.

660 만약 “WIN” 또는 “LOSE”가 되면 게임 끝나며, 그렇지 않으면 oddeven = 1로 돌아가서 구슬의 수를 랜덤으로 다시 설정하며 oddeven = 1, 2, 3, 4를 반복하게 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

(5) 게임 종료 X ( 666 ~ 707 )

666 값이 있다면 게임이 종료되는 ‘end’가 값이 없는 ‘None’일 때의 칠판, 손, 버튼의 이미지는 아래와 같다.

# 칠판 (board)

667 칠판 위치에 칠판 이미지 생성

668 board\_state = 0 : 목표 값(bead\_goal), 획득 값(life)을 포함한 텍스트 설정

670 board\_state = 1 : 구슬의 수(bead\_num), 답(answer) 및 “정답”을 포함한 텍스트 설정

672 board\_state = 2 : 구슬의 수(bead\_num), 답(answer) 및 “오답”을 포함한 텍스트 설정

674 설정된 텍스트를 칠판위에 생성

# 손 (pocket)

676 pocket\_state = None : 손 위치에 문구, 구슬 둘 다 나타나지 않은 손 이미지 생성

678 pocket\_state = 0 : 손 위치에 “홀수? 짝수?” 문구가 포함된 손 이미지 생성

680 pocket\_state = n : 손 위치에 n 개의 구슬이 포함된 손 이미지 생성 ( n : 1 ~ 10 )

# 버튼 (odd, even)

703 홀수 버튼 위치에 “홀수” 이미지 생성

704 Button 클래스를 이용하여 홀수 버튼 위의 마우스 유무에 따라 다른 버튼 이미지로 변경

705 짝수 버튼 위치에 “짝수” 이미지 생성

706 Button 클래스를 이용하여 홀수 버튼 위의 마우스 유무에 따라 다른 버튼 이미지로 변경

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

(6) 게임 종료 O ( 709 ~ 741 )

709 값이 있다면 게임이 종료되는 ‘end’가 “홀수” 또는 “짝수”의 값을 가지므로 게임은 종료된다.

종료하는 과정은 아래와 같다.

710 oddeven = 5 : 1~4 범위의 oddeven은 게임을 진행하는 데 사용되고, 5는 게임을 종료하기 위해 1~4값에서 빠져나오기 위해 사용된다.

711 게임 종료 화면 생성

712 player의 게임 결과 텍스트 설정

713 게임 종료 화면 위에 텍스트 생성

716 게임 결과 텍스트 생성 직후 시간 (end\_time) 측정

721 ( time.time() – end\_time )이 일정 값을 넘어가면 게임 종료 안내 시작

722 ( time.time() – end\_time )이 일정 값이 되기 전에 “잠시 후 게임이 종료됩니다.” 안내

726 ( time.time() – end\_time )이 일정 값이 되기 전에 완전한 게임 종료까지 남은 시간 안내

732 while(running)의 running에 0으로 변환함으로써 while 밖으로 이동

739 반복되는 while문 안에서 display update

741 while문을 빠져나오면 게임 완전히 종료

< 4. 참고 자료 >

1단계 fps 및 캐릭터 이동 작성 참고

* <https://blog.naver.com/realdc33/222480490010>

1단계 스프라이트 작성 참고

* <https://digiconfactory.tistory.com/200>

3단계 Button Class 작성 참고

* <https://m.blog.naver.com/scyan2011/221998190058>

pygame.mouse.get\_pose() 및 pygame.event.poll() 참고

* <https://runebook.dev/ko/docs/pygame/ref/mouse>

< 5.보완할 점 >

1)게임 시작 전 플레이어에게 시작 시점을 알릴 수 있는 수단이 필요하다.

2)캐릭터의 위, 아래 방향으로의 움직임 묘사

3)무궁화 꽃이 피었습니다에서 처럼 줄다리기 게임에서 조금 더 자연스러운 모션 연출