소프트웨어 프로젝트 2

AD project – Rabbit Hood

20191626 오준호

20191634 윤현승

01. 구현 목표 및 사전조사

평소 video 게임을 즐겨 하며 게임이 어떻게 만들어 지는지 궁금하였고, 직접 만들어보고 싶은 흥미도 가지고 있었다.

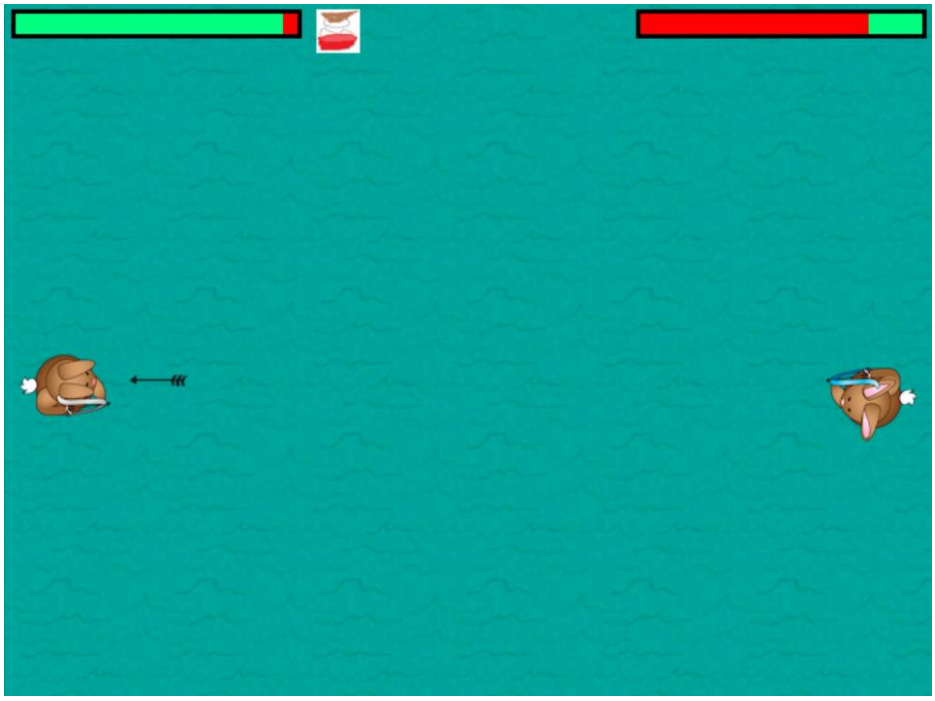
따라서 이번 ADproject 목표는 2P video 게임을 만드는 것이다.

Video 게임을 작성하기 위해서 python 라이브러리 pygame 을 사용하기로 하였다.

게임에 필요한 그래픽과 생동감을 더해줄 사운드 효과들이 필요하여

https://www.raywenderlich.com/2795-beginning-game-programming-for-teens-with-python

에서 리소스들을 가져와 이용하였다.



-구현해야 할 게임의 기능-

* 각 플레이어의 HP 값을 나타내는 HP 바를 구현

* 상대의 HP 값을 감소시키는 화살 구현

* 특정 키를 방향키로 지정 후 플레이어의 위치를 변경

* 플레이어의 HP 값이 0 이 되었을 때, 게임 종료 및 승리 이미지 표시

02. 소프트웨어 구조 설계

```python
# 1 - Import library
import pygame

# 2 - Initialize the game
pygame.init()
width, height = 640, 480
screen=pygame.display.set_mode((width, height))

# 3 - Load images
player = pygame.image.load("resources/images/dude.png")

# 4 - keep looping through
while 1:
    # 5 - clear the screen before drawing it again
    screen.fill(0)
    # 6 - draw the screen elements
    screen.blit(player, (100,100))
    # 7 - update the screen
    pygame.display.flip()
    # 8 - loop through the events
    for event in pygame.event.get():
        # check if the event is the X button
        if event.type==pygame.QUIT:
            # if it is quit the game
            pygame.quit()
            exit(0)
```

1.필요한 라이브러리를 가져오는 기능

2.게임 실행 시 필요한 여러 값들을 초기화 시켜주는 생성자 역할을 하는 기능

3.필요 이미지를 불러오는 기능

4.실제 게임이 구동되면서 계속해서 event 값을 처리하는 기능

기능에 따라서 구역을 나누어 게임 소프트웨어의 큰 구조를 만들었다.

03. 구현 상세 설계 및 코딩

```
# 1 - Import library
import pygame
from pygame.locals import *
import random
```

게임 GUI 구성을 위해 Pygame 라이브러리를 가져왔다.

Random 모듈은 게임에서 **화살의 데미지**와 **포션회복량**을 일정한 값이 아니라 랜덤으로 적용시키기 위해 가져왔다.

```
# 2 - Initialize the game
pygame.init()
width, height = 640, 480
screen=pygame.display.set_mode((width, height))
key1 = [False, False]
playerpos1 = [10, height // 2]
key2 = [False, False]
playerpos2 = [width - 75, height // 2]
arrow1_xy = []
arrow2_xy = []
healthvalue_1 = 194
healthvalue_2 = 194
pygame.mixer.init()
p1_potion_cnt = 1
p2_potion_cnt = 1
```

Pygame 의 생성자를 실행시키고, 사용자에게 보여지는 화면의 크기를 정해주었다.

player1 과 player2 의 방향키의 keydown(키가눌린상태)과 keyup(키를뗀상태)을 확인하기 위하여 Key1, key2 boolean 값을 가진 리스트를 만들어주었다.

초기 player 의 위치를 알려주는 좌표값 playerpos1 과 playeros2 를 만든다.

arrow1_xy, arrow2_xy 는 frame 에 존재하는 화살을 발사된 순서로 [x 좌표, y 좌표]를 저장한 2 차원형태의 리스트이다.

각 player 의 초기 healthvalue 값을 동일하게 초기화 시켜주었다.

```
# 3 - Load images
player1 = pygame.image.load("resources/images/dude.png")
player2 = pygame.image.load("resources/images/dude2.png")
player2 = pygame.transform.rotate(player2, 180)
grass = pygame.image.load("resources/images/grass.png")
arrow = pygame.image.load("resources/images/bullet.png")
re_arrow = pygame.transform.rotate(arrow, 180)
potion = pygame.image.load("resources/images/potion.png")
FPS = 70
fpsClock = pygame.time.Clock()
healthbar = pygame.image.load("resources/images/healthbar.png")
health = pygame.image.load("resources/images/health.png")
player1_win = pygame.image.load("resources/images/player1_win.png")
player2_win = pygame.image.load("resources/images/player2_win.png")
# 3.1 - Load audio
hit = pygame.mixer.Sound("resources/audio/explode.wav")
shoot = pygame.mixer.Sound("resources/audio/shoot.wav")
hit.set_volume(0.05)
shoot.set_volume(0.05)
pygame.mixer.music.load('resources/audio/moonlight.wav')
pygame.mixer.music.play(-1, 0.0)
pygame.mixer.music.set_volume(0.25)
```

게임에 필요한 player, grass, arrow.....등등 이미지를 load 하였다.

Player2 는 이미지를 가져온 후 player1 과 마주 보게 만들기 위해서 pygame 의 메소드인 transform.rotate 를 사용하여 인자값으로 180 을 주어 180 를 돌렸다.

화살을 발사했을 때, 화살에 맞았을 때 사용할 사운드를 가져와 볼륨을 적절한 값으로 설정하여 변수에 저장해주었다.

Pygame 의 배경음악을 설정하여 play 해주었다. 이때, 인자값으로 (-1, 0.0) 루프값은 -1 이므로 음악이 무한정 반복되게 시작값은 0.0 으로 음악이 재생 중이었다면 다시 시작되게 설정해주었다.

```
# 4 - keep looping through
while 1:
    # 5 - clear the screen before drawing it again
    screen.fill(0)
    # 6.1 - draw the screen elements
    for x in range(width//grass.get_width()+1):
        for y in range(height//grass.get_height()+1):
            screen.blit(grass,(x*100,y*100))

    # 6.2 - Draw arrows
    if len(arrow1_xy) != 0:
        for ax,ay in arrow1_xy:
            screen.blit(arrow, (ax,ay))
    if len(arrow2_xy) != 0:
        for ax,ay in arrow2_xy:
            screen.blit(re_arrow, (ax,ay))
```

While 문 이후 코드는 게임을 구동했을 때 핵심적인 event 를 처리하고 각 frame 마다 조건에 맞게 image 를 그려 움직이는 것처럼 보여주는 코드이다.

조건에 맞게 현재 frame 의 image 를 그리기 전에 전 frame 이미지를 screen.fill(0)을 통해 검은색 픽셀로 채운 빈 이미지로 만들어준다.

만들어진 검은색의 빈 이미지에 개임 배경 화면인 100 X 100 픽셀크기의 잔디 이미지를 이미지의 너비와 높이에 맞게 for 문으로 그려 채워주었다.

len(arrow1_xy) != 0 로 화살이 존재하는지 확인한 후, 화살이 존재한다면 for 문을 통해 화살들을 좌표값에 그려준다.

```
# 6.3.2 - Check for collisions
index1 = 0
for bullet in arrow1_xy:
    player2rect = pygame.Rect(player2.get_rect())
    player2rect.left = playerpos2[0]
    player2rect.top = playerpos2[1]
    bullrect = pygame.Rect(arrow.get_rect())
    bullrect.left = bullet[0]
    bullrect.top = bullet[1]
    if player2rect.colliderect(bullrect):
        hit.play()
        arrow1_xy.pop(index1)
        print('아야2')
        healthvalue_2 -= random.randint(8,20)
        if healthvalue_2 < 0:
            healthvalue_2 = 0
    index1 += 1
```

화살과 player 가 충돌함을 확인하는 코드이다.

Pygame 의 Rect 메소드를 이용하여 화살과 플레이어를 네모 박스 이미지로 변환시켜 변수에 저장한다.

Rect 이 지원하는 colliderect 메소드를 이용하여 충돌을 확인하여 충돌했을 경우 화살을 그 화살을 arrow_xy 리스트에서 pop 을 통해 삭제하고, 맞은 player 의 healthvalue 의 값을 감소시켰다.

```
# 6.5 - Draw health bar
screen.blit(healthbar, (5,5))
for health1 in range(healthvalue_1):
    screen.blit(health, (health1+8,8))
screen.blit(healthbar, (width - 205, 5))
for health1 in range(width - 1, (width - 1) - healthvalue_2, -1):
    screen.blit(health, (health1 - 8, 8))

# 6.6 - Draw potion
if p1_potion_cnt != 0:
    screen.blit(potion, (215, 5))
if p2_potion_cnt != 0:
    screen.blit(potion, (width - 240, 5))

# Draw player
screen.blit(player1, playerpos1)
screen.blit(player2, playerpos2)


# 7 - update the screensw
pygame.display.flip()
fpsClock.tick(FPS)
```

빨간색 픽셀로 이루어진 healthbar 이미지를 양쪽 플레이어 위쪽에 그린다

그후 healthbar 와 높이 픽셀은 같지만 가로 가 얇은 초록색 픽셀로 이루어진 health 이미지를 player 가 가진 healthvalue 값에 비례하게 for 문으로 채운다.

포션 이미지는 사용할 포션이 0 이 아닐 때만 그리도록 설정하였다.

player 들을 playerppos 좌표에 그린다.

Pygame 의 디스플레이를 업데이트 한다.

초당 프레임 값(FPS)를 설정한다.

```
# 8 - loop through the event
for event in pygame.event.get():
    # check if the event is the X button
    if event.type==pygame.QUIT:
        # if it is quit the game
        pygame.quit()
        exit(0)
    if event.type == pygame.KEYDOWN:
        print(event.key)
        #shoot arrow
        if event.key==pygame.K_SPACE:
            shoot.play()
            arrow1_x = playerpos1[0]
            arrow1_y = playerpos1[1] + 30
            arrow1_xy.append([arrow1_x,arrow1_y])
        if event.key==pygame.K_SLASH:
            shoot.play()
            arrow2_x = playerpos2[0]
            arrow2_y = playerpos2[1] + 5
            arrow2_xy.append([arrow2_x,arrow2_y])
        #drink potion
        if event.key==pygame.K_q and p1_potion_cnt > 0:
            healthvalue_1 += random.randint(30, 100)
            if healthvalue_1 > 194:
                healthvalue_1 = 194
            p1_potion_cnt -= 1
```

게임 실행 중 발생하는 event 를 for 문으로 반복적으로 받아온다.

이때 event 의 타입이 pygame.KEYDOWN 일 때 즉 키가 눌렸을 때, 각 키의 상호작용에 맞게 처리하도록 만든 코드이다.

화살발사 키가 눌리면 가져왔던 화살발사 사운드를 실행시키고, array_xy 에 화살을 추가하며 좌표 값은 현재 player 의 활로 정한다.

포션키가 눌리면 포션 카운트를 감소하고 healthvalue 값을 30~100 사이값으로 랜덤으로 증가시킨다.

```
    #player1's keydown
    if event.key==K_w:
        key1[0]=True
    elif event.key==K_s:
        key1[1]=True
    #player2's keydown
    elif event.key==K_UP:
        key2[0]=True
    elif event.key==K_DOWN:
        key2[1]=True

if event.type == pygame.KEYUP:
    #player1's keyup
    if event.key==pygame.K_w:
        key1[0]=False
    elif event.key==pygame.K_s:
        key1[1]=False

    #player2's keyup
    elif event.key==pygame.K_UP:
        key2[0]=False
    elif event.key==pygame.K_DOWN:
        key2[1]=False
```

```
# 9.1 - Move player1
if key1[0]:
    if playerpos1[1] > 50:
        playerpos1[1]-=5
elif key1[1]:
    if playerpos1[1] < height - 60:
        playerpos1[1]+=5

# 9.2 - Move player2
if key2[0]:
    if playerpos2[1] > 50:
        playerpos2[1]-=5
elif key2[1]:
    if playerpos2[1] < height - 60:
        playerpos2[1]+=5

# 9.3 - move arrow
if len(arrow1_xy) != 0:
    for i, axy in enumerate(arrow1_xy):
        axy[0] += 20

        arrow1_xy[i][0] = axy[0]
        if axy[0] >= width:
            arrow1_xy.remove(axy)
```

player 들의 이동키가 keydown 되었을 때는 key 리스트의 false 값을 true 값으로 바꿔주어 if 문을 통해 true 이면 player 가 그려지는 위치 좌표 값인 playerpos 를 이동시켜준다. 이때, playerpos 좌표값이 화면을 밖의 좌표 값을 잡지 않도록 if 문으로 방지했다.

If 조건 = len(arrow_xy)를 통해 화살의 존재를 확인한 후에, 존재한다면 화살의 위치 값 중 x 좌표의 값을 player 가 쏘는 방향에 맞게 + 또는 – 방향으로 특정 값만큼 픽셀을 이동해준다.

```
#10 - Winner check
if healthvalue_1 == 0:
    running = 0
    winner = 2
    break
elif healthvalue_2 == 0:
    running = 0
    winner = 1
    break
```

```
# 11 - Win/lose display
if winner == 2:
    pygame.mixer.music.stop()
    screen.blit(player2_win, (0,0))

elif winner == 1:
    pygame.mixer.music.stop()
    screen.blit(player1_win, (0,0))

while 1:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit(0)
    pygame.display.flip()
```

player 들의 healthvalue 값을 확인하여 0 이라면 이긴 플레이어를 확인하도록 winner 변수에 값을 대입한후 while 문을 break 하며 게임 실행을 중단시킨다.

게임이 종료된 후에 winner 값을 확인하여 winner 에 맞는 이미지를 띄워준다

04. 개선할 점

처음 프로그램을 설계하는 단계에서는 간단한 프로그램이라서, 여러 개의 모듈로 구성하는 것이 아니라 한 파일에서 순환문의 로직에 따라서 실행되도록 만들었다. 이렇게 하면 다음에 이 게임을 확장할 때 아쉬운 부분이 있었다. 현재는 게임 플레이를 한 번 밖에 하지 못한다. 게임이 끝나고 새로운 게임을 시작하려면 진행되고 있는 프로그램을 종료하고, 새롭게 실행을 해야 한다. 모듈화를 통해서, 혹은 꼭 모듈화를 사용하지 않더라도 메인 화면에서 게임 로직으로 들어갈 수 있도록 하고 한 번 게임이 끝나면 다시 메인 화면으로 넘어가도록 만들 수도 있었을 것이다. 하지만 처음 설계했던 부분대로 프로젝트를 진행하였고, 시간이 많지 않아서 예상하기 힘들었던 모듈화 부분을 하지 못한 것이 가장 아쉽다.

그리고 처음 설계한 방식(순환문의 로직에 따라서 게임이 진행되는 방식)대로 진행을 하면, 따로 메소드가 필요한 것이 없기 때문에, Unit Test 를 활용한 단위 테스트를 작성하는 것이 거의 불가능 했다. 그래서 단위 테스트를 작성하는 것 대신, 프로그램을 작성하는 중간 중간에 화살이 엔진에 잘 맞는지 확인하기 위해서 print 문을 사용하여 checking 을 하는 방법을 채택하여 테스트를 진행하였다.