-소프트웨어 융합개론 project3-

데이터 사이언스를 활용한 게임 '이터널리턴'의 가장 효율적인 파밍 루트 분석 최종 보고서

2023105420 김태현

목차

프로젝트 개요 및 목표

이터널리턴이란?

'이터널 리턴'은 님블뉴런사에서 만든 쿼터 뷰 배틀로얄 게임이다. 배틀로얄 게임이기 때문에 아이템이 중요하다. 또한 '이터널리턴'에는 '무기숙련도'라는 시스템이 존재하는데 아이템 만큼이나 캐릭터 능력치에 미치는 영향이 크다. 무기 숙련도를 올리기 위해선 게임 속 야생동물이나 플레이어와의 전투가 필요하기 때문에 빠른 아이템 파밍이 중요하다. 빠른 파밍에 가장 큰 영향을 미치는 것은 파밍루트이다. 하지만 캐릭터마다 사용하는 아이템이 다르기 때문에, 파밍루트는 획일화 되어있지 않으며, 지역마다 나오는 재료들이 다르기 때문에 직관적으로 파악하기 어렵다. 이러한 문제해결을 위해 게임 정보 사이트를 파이썬 라이브러리인 selenium을 통해 크롤링 후 분석하여, 캐릭터별 아이템 조합을 찾고, 그에 맞는 최적의 파밍루트를 찾는 프로젝트를 진행해보고자 한다

프로젝트의 의의

'이터널리턴'게임 내에도 루트 제작 시스템이 존재한다. 그러나 아이템 별 최적 루트를 찾아주는 시스템이지 최적루트를 직접 만들어주는 시스템이 아니다. 따라서 아이템을 하나씩 직접 골라야 하며 직관적으로 효율을 확인할 수 없다는 한계가 존재한다. 그리고 원격 드론 호출 재료 즉, 구매 재료를 시스템에서 정해주는 것이 아닌 직접 골라야한다는 단점 또한 존재한다. 따라서 인게임 시스템의 한계를 극복해 보다 더 효율적인 루트를 짤 수 있는 시스템을 만들고 이를 시각화하여 좀더 직관적으로 확인할 수 있게 하는 것을 프로젝트의 최종 목표로 한다

[그림 1] 인게임 루트 제작 화면

프로젝트 구성

사용 언어 – python

사용라이브러리 – selenium, re, urllib.request, os, tkinter, functools, PIL

프로젝트 기본 설계

1단계 - python 라이브러리인 selenium을 통하여 웹에서 데이터 크롤링

2단계 – 크롤링한 데이터 체계적 정리

3단계 – 프로젝트 핵심 기능인 최적 루트 산출 알고리즘 고안

4단계 – 1,2,3 단계 데이터를 병합해 시각화

프로젝트 세부내용

1. selenium을 활용한 데이터 크롤링

파이썬 라이브러리인 selenium과 urllib.request 을 통해 이터널리턴 인벤 이터널리턴 캐릭터 데이터베이스 (https://er.inven.co.kr/db/character)와 아이템 데이터베이스 (https://er.inven.co.kr/db/item)에서 각각 캐릭터 정보/이미지, 아이템 정보/이미지를 크롤링 하였다. 그 후 받아온 정보를 가공해 regex를 활용해 txt, png파일로 저장하였다.

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By

import time
import re

options = Options()
options.add_experimental_option('detach', True)
options.add_experimental_option('excludeSwitches', ['enable-logging'])
```

필요한 라이브러리와 기본 옵션 설정

```
def get_data():

    wd_item = webdriver.Chrome(options=options)
    wd_item.get("https://er.inven.co.kr/db/item#r4c3")
    time.sleep(1)

    item_info = dict()

    for i in range(1,185):

        item_name = wd_item.find_element(By.CSS_SELECTOR, "tbody > tr:nth-child("+str(i)+") > td:nth-child(2) > span > a").get

        item_case = wd_item.find_element(By.CSS_SELECTOR, "tbody > tr:nth-child("+str(i)+") > td.class2text.txt_center > span"

        if item_case != "재료":

            item_option = wd_item.find_elements(By.CLASS_NAME,"option_text")
            item_option = item_option[3*i-1].get_attribute('innerHTML')
            p = re.compile('(?=>)(.*?)(?=<)')
            p_list = p.findall(item_option)

            for s in range(len(p_list)):

                p_list[s] = p_list[s][1:]

            item_option = p_list

            if '(고유)' in item_option:
                item_option.pop(item_option.index('(고유)'))

            item_option = list(filter(None, item_option))

        else:
            item_option = ['재료']


        item_matarial_adress = wd_item.find_element(By.CSS_SELECTOR,"tbody > tr:nth-child("+str(i)+") > td:nth-child(2) > span
        mat_ad = webdriver.Chrome(options=options)
        mat_ad.get(item_matarial_adress)
        item_matarial_list = mat_ad.find_elements(By.CLASS_NAME,"itemname")
        item_matarial_core = []

        for t in item_matarial_list:

            if t.get_attribute('data-color-itemgrade') == "1":
                item_matarial_core.append(t.get_attribute("innerHTML"))

        mat_ad.close()
```

아이템 정보를 CSS_SELECTOR/CLASS_NAME을 이용해 크롤링 후 item_info dictionary에 저장

```python
def item_locate():

    base_item_dv = webdriver.Chrome(options=options)
    base_item_dv.get("https://er.inven.co.kr/db/item#r4c0")
    time.sleep(1)

    base_item_dict = dict()

    for i in range(1,85):
        base_item = base_item_dv.find_element(By.CSS_SELECTOR,"tr:nth-child("+str(i)+") > td:nth-child(2) > span > a").get_att
        base_item_loc = base_item_dv.find_element(By.CSS_SELECTOR,"tr:nth-child("+str(i)+") > td:nth-child(5) > span").get_att

        p = re.compile('(?=발견 장소)(.*?)(?=<)')
        q = re.compile('(?<= )(.*?)(?=\()')
        r = re.compile("'([^']+)'")
        base_item_loc_1st = p.findall(base_item_loc)
        base_item_loc_2nd = str(q.findall(str(base_item_loc_1st))).replace("장소: ","")
        base_item_loc_3rd = r.findall(base_item_loc_2nd)

        base_item_dict[base_item] = base_item_loc_3rd

    base_item_dv.close()

    f_bid = open("base_item_dict.txt","w")

    for i in base_item_dict.keys():
        f_bid.write(i+"\n")

        for s in base_item_dict[i]:
            f_bid.write(s+" ")
        f_bid.write("\n")

    f_bid.close
```

재료를 CSS_SELECTOR를 통해 크롤링 후 base_item dictionary에저장

```python
def get_data():

    wd_char = webdriver.Chrome(options=options)
    wd_char.get("https://er.inven.co.kr/db/character")
    time.sleep(1)

    final_save = dict()

    links = wd_char.find_elements(By.CLASS_NAME,"layer_detail")
    char_info = open("char_info.txt","w")
    for i in range(int(len(links)/3)):
        temp_list = []
        link = links[3*i].get_attribute("href")

        char_info_wd = webdriver.Chrome(options=options)
        char_info_wd.get(link)
        time.sleep(2)
        charactor = char_info_wd.find_element(By.CLASS_NAME,'skinName').get_attribute("innerHTML")
        weapon_raw = char_info_wd.find_elements(By.CSS_SELECTOR,'div.skillTitle > span:nth-child(3)')

        for s in weapon_raw:
            temp_list.append(s.get_attribute("innerHTML")[9:-1])

        temp_list2 = copy.deepcopy(temp_list)
        final_save[charactor] = temp_list2

        char_info_wd.close()

    wd_char.close()
```

캐릭터 정보를 CLASS_NAME을 통해 크롤링 후 dictionary에 저장

```
f_bid = open("base_item_dict.txt","w")

for i in base_item_dict.keys():
    f_bid.write(i+"\n")

    for s in base_item_dict[i]:
        f_bid.write(s+" ")
    f_bid.write("\n")

f_bid.close
```

```
f_if = open("item_info.txt","w")

for i in item_info.keys():
    f_if.write(i+"\n")

    for s in item_info[i]:
        f_if.write(str(s)+" ")
        f_if.write("\n")

f_if.close
```

```
wd_char.close()

for i in final_save.keys():
    char_info.write(i+"\n")

    for s in final_save[i]:
        char_info.writelines(s)
    char_info.write("\n")

char_info.close()
```

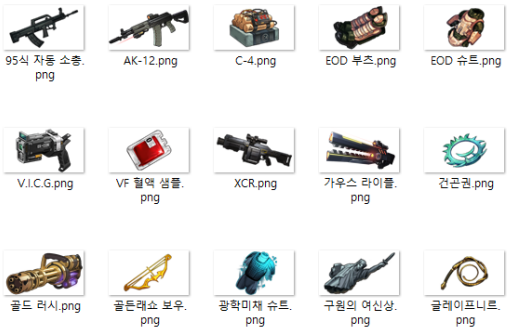
dictionary에 저장했던 정보를 각각 txt파일로 저장

```
def char_data():

    char_img = webdriver.Chrome(options=options)
    char_img.get("https://er.inven.co.kr/db/character")
    time.sleep(1)

    if not os.path.exists("./char_img"):
        os.makedirs("./char_img")

    for i in range(1,70):
        img = char_img.find_element(By.CSS_SELECTOR," div > div:nth-child("+str(i)+") > div.db_block.card_img > a:nth-child(2)
        img_name = char_img.find_element(By.CSS_SELECTOR,"div > div:nth-child("+str(i)+") > div.db_block.card_info > p > a:nth
        f = open("./char_img/"+img_name+".jpg",'wb')
        f.write(urllib.request.urlopen(img).read())
        f.close()

    char_img.close()

def item_data():

    item_img = webdriver.Chrome(options=options)
    item_img.get("https://er.inven.co.kr/db/item#r4c3")
    time.sleep(1)

    if not os.path.exists("./f_item_img"):
        os.makedirs("./f_item_img")

    for i in range(1,185):
        img = item_img.find_element(By.CSS_SELECTOR,"table > tbody > tr:nth-child("+str(i)+") > td.option_text.txt_center > sp
        img_name = item_img.find_element(By.CSS_SELECTOR,"table > tbody > tr:nth-child("+str(i)+") > td:nth-child(2) > span >
        f = open("./f_item_img/"+img_name+".jpg",'wb')
        f.write(urllib.request.urlopen(img).read())
        f.close()

    item_img.close()

def matarial_data():

    matarial_img = webdriver.Chrome(options=options)
    matarial_img.get("https://er.inven.co.kr/db/item#r4c0")
    time.sleep(1)

    if not os.path.exists("./matarial_img"):
        os.makedirs("./matarial_img")

    for i in range(1,85):
        img = matarial_img.find_element(By.CSS_SELECTOR,"table > tbody > tr:nth-child("+str(i)+") > td.option_text.txt_center
        img_name = matarial_img.find_element(By.CSS_SELECTOR,"table > tbody > tr:nth-child("+str(i)+") > td:nth-child(2) > spa
        f = open("./matarial_img/"+img_name+".jpg",'wb')
        f.write(urllib.request.urlopen(img).read())
        f.close()

    matarial_img.close()
```

캐릭터/아이템/재료 이미지를 scr 요소를 통해 크롤링 후 png파일로 저장

```
95식 자동 소총
돌격소총
['공격력 +72', ' 기본 공격 사거리 +0.5']
['페도로프 자동소총', '화약', '피아노선', '분필', '돌멩이']
AK-12
돌격소총
['공격력 +74', '치명타 확률 +22%']
['페도로프 자동소총', '화약', '피아노선', '유리병', '돌멩이', '접착제']
C-4
설치
['트랩 피해 +400']
['쥐덫', '화약', '배터리', '피아노선']
EOD 부즈
다리
['방어력 +8', '이동 속도 +0.27', ' 방해 효과 저항 +20%']
['운동화', '가죽', '타이즈', '접착제', '고철']
EOD 슈트
옷
['방어력 +22', '레벨 당 최대 체력 +14~280', '체력 재생 +60%', '쿨다운 감소 +15%']
['바람막이', '나뭇가지', '고철', '승복', '붕대']
NTW-20
저격총
['공격력 +51', '스킬 증폭 +79', '시야 +1']
['화승총', '원석', '유리병', '돌멩이', '접착제']
```

[그림 2] txt 파일 예시



[그림3] png 파일 예시

2. 그리디 알고리즘을 이용한 최적 루트 산출

앞서 크롤링 한 데이터를 바탕으로 생성된 txt 파일을 불러온 후 dictionary와 class를 활용해 데이터를 정리하였다. 이를 바탕으로 캐릭터별 가능한 아이템 조합을 생성하고 그리디 알고리즘을 활용해 가장 적은 수의 루트를 통해 파밍을 할 수 있게 하는 핵심 기능을 구현하였다.

```python
class Item:

    def __init__(self, name, type, options, matarial):

        self.name = name
        self.type = type
        self.options = options
        self.matarial = matarial

class Charactor:

    def __init__(self, name, weapon, power):

        self.name = name
        self.weapon = weapon
        self.power = power
```

아이템과 캐릭터 정보를 저장할 class

```
item_info = open("item_info.txt","r")
item_info_r = item_info.readlines()

item_info_dict = dict()

i_name = re.compile("(?=\w)(.*?)\n")
i_type = re.compile("(?=\w)(.*?) \n")
i_options = re.compile("(?=\w)(.*?)\'")

for i in range(int(len(item_info_r)/4)):

    item_name = i_name.findall(item_info_r[4*i])[0]
    item_type = i_type.findall(item_info_r[4*i+1])[0]
    item_options = i_options.findall(item_info_r[4*i+2])
    item_matarial = i_options.findall(item_info_r[4*i+3])
    item_info_dict[item_name] = Item(item_name, item_type, item_options, item_matarial)

item_info.close()
```

아이템 정보를 regex를 통해 dictionary를 통해 class를 불러오는 형태로 데이터베이스화

```
char_info = open("char_info.txt","r")
char_info_r = char_info.readlines()

char_info_dict = dict()

c_name = re.compile("(?=\w)(.*?)\n")
c_weapon = re.compile("(?=\w)(.*?)\(.*?\)")
c_power = re.compile("(?<=\()(.*?)\)")

for i in range(int(len(char_info_r)/2)):
    charactor_name = c_name.findall(char_info_r[2*i])[0]
    charactor_weapon = c_weapon.findall(char_info_r[2*i+1])
    charactor_power = c_power.findall(char_info_r[2*i+1])
    char_info_dict[charactor_name] = Charactor(charactor_name, charactor_weapon, charactor_power)

char_info.close()
```

캐릭터 정보를 regex를 통해 dictionary를 통해 class를 불러오는 형태로 데이터베이스화

```
base_item = open("base_item_dict.txt","r")
base_item_r = base_item.readlines()
r_name = re.compile("(?=\D)(.*?)\n")
r_place = re.compile("(?=\D)(.*?)\W")

matarial_dict = dict()

for i in range(int(len(base_item_r)/2)):
    matarial_dict[r_name.findall(base_item_r[2*i])[0]] = r_place.findall(base_item_r[2*i+1][:-1])

base_item.close()
```

재료를 regex를 활용하여 dictionary 구조로 데이터베이스화

```python
def choose_item(char_name,a):

    global weapon_list, head_list, clothes_list, arm_list, leg_list

    index_weapon = char_info_dict[char_name].weapon.index(a)
    ability_setting = char_info_dict[char_name].power[index_weapon]

    weapon_list = []
    head_list = []
    clothes_list = []
    arm_list = []
    leg_list = []

    if ability_setting == "스킬증폭":

        for i in item_info_dict.keys():

            if item_info_dict[i].type == a and "스킬 증폭" in str(item_info_dict[i].options):
                weapon_list.append(item_info_dict[i])
            if ("스킬 증폭" or "쿨다운 감소") in str(item_info_dict[i].options):
                if not ("공격력" or "치명타 확률" or '레벨 당 공격력') in str(item_info_dict[i].options):
                    if item_info_dict[i].type == "머리":
                        head_list.append(item_info_dict[i])
                    if item_info_dict[i].type == "옷":
                        clothes_list.append(item_info_dict[i])
                    if item_info_dict[i].type == "팔":
                        arm_list.append(item_info_dict[i])
                    if item_info_dict[i].type == "다리":
                        leg_list.append(item_info_dict[i])

    elif ability_setting == "스증방어력":

        for i in item_info_dict.keys():
            if item_info_dict[i].type == a and "스킬 증폭" in str(item_info_dict[i].options):
                weapon_list.append(item_info_dict[i])
            if ("스킬 증폭" or "쿨다운 감소" or "방어력" or "최대 체력") in str(item_info_dict[i].options):
                if not ("공격력" or "치명타 확률" or '레벨 당 공격력') in str(item_info_dict[i].options):
                    if item_info_dict[i].type == "머리":
                        head_list.append(item_info_dict[i])
                    if item_info_dict[i].type == "옷":
                        clothes_list.append(item_info_dict[i])
                    if item_info_dict[i].type == "팔":
                        arm_list.append(item_info_dict[i])
                    if item_info_dict[i].type == "다리":
                        leg_list.append(item_info_dict[i])
```

위에서 만든 데이터베이스를 바탕으로 최적의 아이템 조합 생성

```python
def item_loop(weapon,limit):
    buy_item = []
    buy_num_list = []
    temp_item = []
    temp_route = []
    temp_shortest = 5

    for head in head_list:
        for clothes in clothes_list:
            for arm in arm_list:
                for leg in leg_list:
                    buy_list, buy_num, route, route_length = route_algorithm([weapon,head.name,clothes.name,arm.name,leg.name]

                    if route_length < temp_shortest:
                        temp_item = []
                        temp_route = []
                        buy_item = []
                        buy_num_list = []
                        buy_item.append(buy_list)
                        buy_num_list.append(buy_num)
                        temp_item.append([item_info_dict[weapon],head,clothes,arm,leg])
                        temp_route.append(route)
                        temp_shortest = route_length
                    elif route_length == temp_shortest:
                        buy_item.append(buy_list)
                        buy_num_list.append(buy_num)
                        temp_item.append([item_info_dict[weapon],head,clothes,arm,leg])
                        temp_route.append(route)
                    else:
                        pass
```

아이템 조합과 루트를 저장

더 짧은 루트가 나오면 삭제 후 그와 같거나 짧은 길이의 루트만 저장하는 로직

```python
def place_algorithm(mat_dict):

    for i in mat_dict.keys():
        for s in matarial_dict[i]:
            if s in place_dict.keys():
                place_dict[s] += 1
            else:
                place_dict[s] = 1


    for i in place_dict.keys():
        if place_dict[i] == max(place_dict.values()):
            max_place = i

    route_list.append(max_place)

    del_list = []

    for i in mat_dict.keys():
        if max_place in matarial_dict[i]:
            del_list.append(i)

    for i in del_list:
        del(mat_dict[i])
```

place_algorithm :가장 많은 종류의 재료를 얻을 수 있는 장소를 현재해로 설정

```python
def route_algorithm(choosen,limit):

    global place_dict, route_list

    mat_list = list()
    mat_dict = dict()
    route_list = list()

    for i in choosen:
        for s in item_info_dict[i].matarial:
            mat_list.append(s)

    for i in mat_list:

        if i in mat_dict.keys():
            mat_dict[i] += 1
        else:
            mat_dict[i] = 1

    trash_list = ["가죽","나뭇가지","돌멩이"]
    for i in trash_list:
        if i in mat_dict:
            del(mat_dict[i])

    route_length = 1

    while bool(mat_dict) == True:
        place_dict = dict()
        place_algorithm(mat_dict)

        if sum(mat_dict.values()) <= limit:
            buy_mat_list = list()
            for i in mat_dict.keys():
                buy_mat_list.append(i)
            return buy_mat_list, len(mat_dict.values()), route_list, route_length

        else:
            route_length += 1
```

구매 가능 재료수를 limit이라는 변수에 받고 남은 재료 개수가 limit보다 작다면 루트를 저장하고 아니면 route_length에 1을 더한 후 다시 place_algorithm 함수를 작동시킴

3. tkinker을 이용한 데이터 정보 시각화

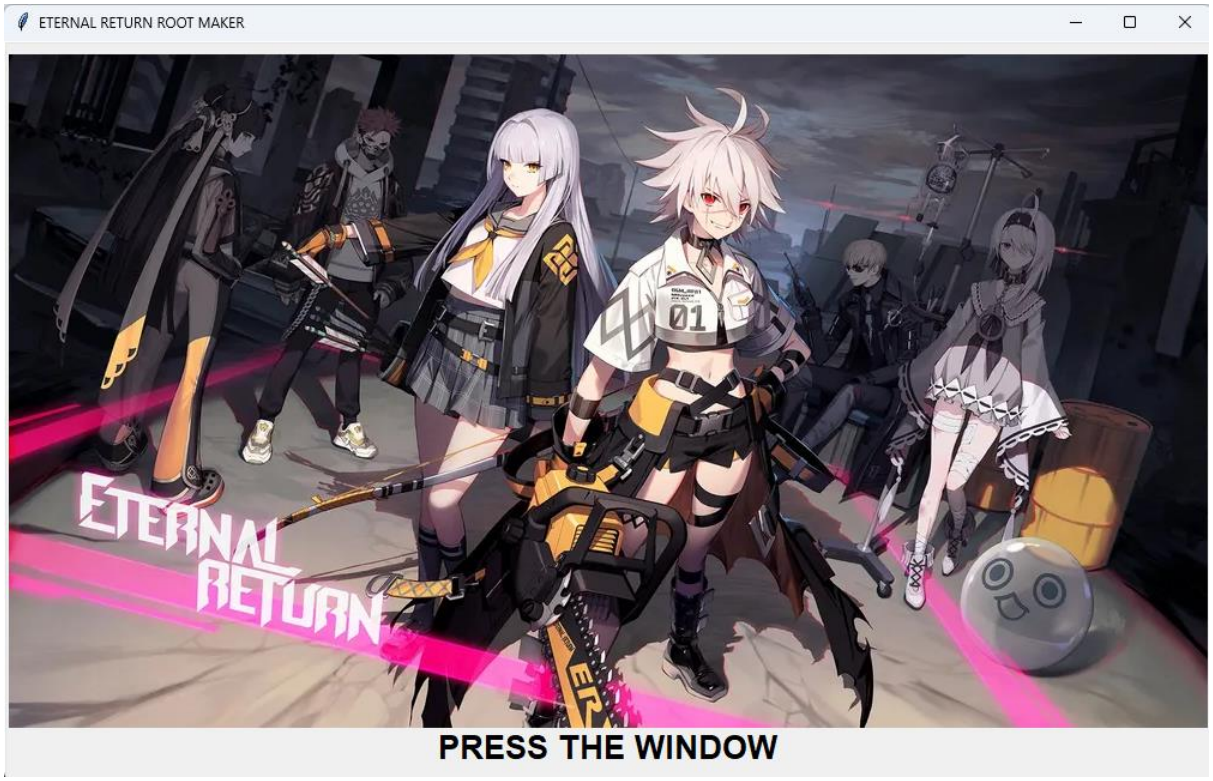
tkinter 라이브러리를 통해 크롤링한 데이터와 핵심 루트 제작 기능을 GUI하여 직관적으로 확인 가능하게 하였다

```
def start():

    global matarial_dict, item_info_dict, char_info_dict

    matarial_dict, item_info_dict, char_info_dict = setting()

    Char_choice()


def Char_choice():
    global Char_Frame

    char_list = os.listdir("./char_img")

    btname_dict = dict()

    for i in char_list:
        btname_dict[i[:-4]] = Char_bt(i[:-4],"./char_img/"+i, "")

    Main.destroy()

    Char_Frame = Tk()
    Char_Frame.title("ETERNAL RETURN ROOT MAKER")
    Char_Frame.geometry("+200+200")

    x_count = 0
    y_count = 0
    count = 0

    photo_list = []
    partial_list = []
    name_list = []

    for i in btname_dict:
        photo_list.append(PhotoImage(file=str(btname_dict[i].pngname)))
        partial_list.append(partial(Weapon_choice,(btname_dict[i].name)))
        name_list.append(btname_dict[i].name)

    for i in btname_dict:

        btname_dict[i].button = Button(Char_Frame, height=120, width=105, overrelief="solid", command=partial_list[count]
                                       ,image=photo_list[count])

        btname_dict[i].button.grid(row=y_count*111,column =x_count*111)

        count += 1
        if x_count == 8:
            x_count = 0
            y_count += 1
        else:
            x_count += 1
        if count == 68:
            break

    Char_Frame.mainloop()
```

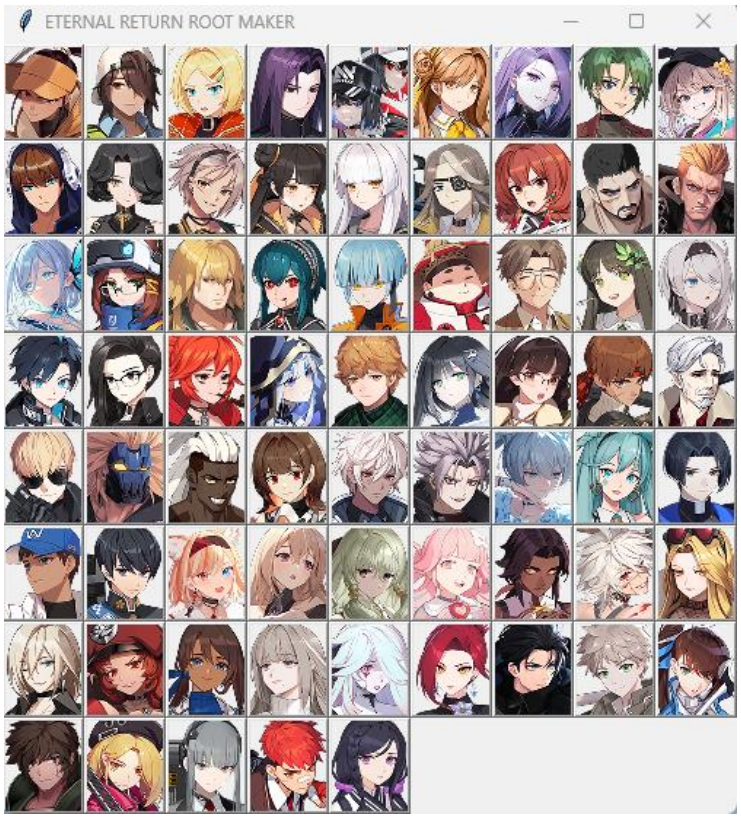
미리 구현해 놓은 루트 구현 기능을 main.py를 import 하여 불러왔다

버튼을 통해 현 프레임이 닫히는 동시에 다음 프레임이 열리게 하여 페이지를 넘어가는 기능을 구현했다

partial을 이용해 tkinter의 command 기능에 인자를 받는 기능을 구현하였다

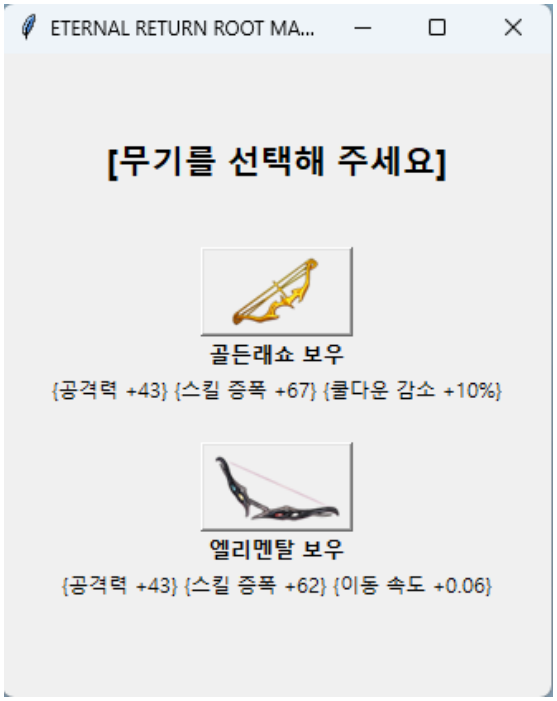


메인화면 : 창을 누르면 캐릭터 선택창으로 이동

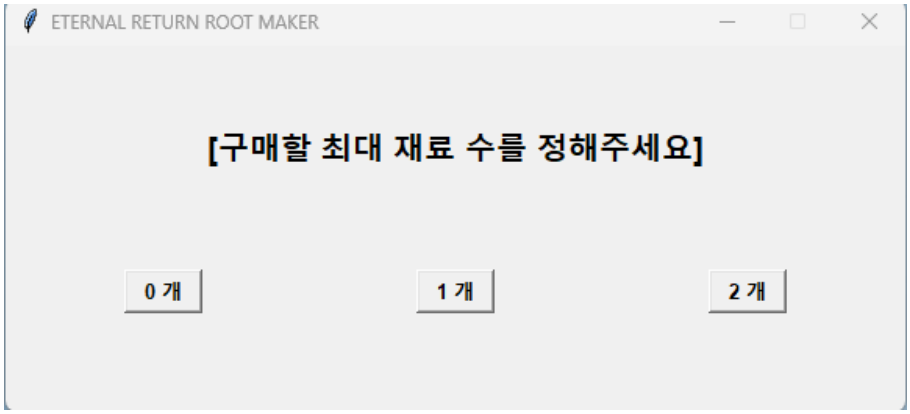


캐릭터 선택화면 : 선택가능한 캐릭터 표시 / 캐릭터를 선택하면 무기 종류 선택창으로 이동

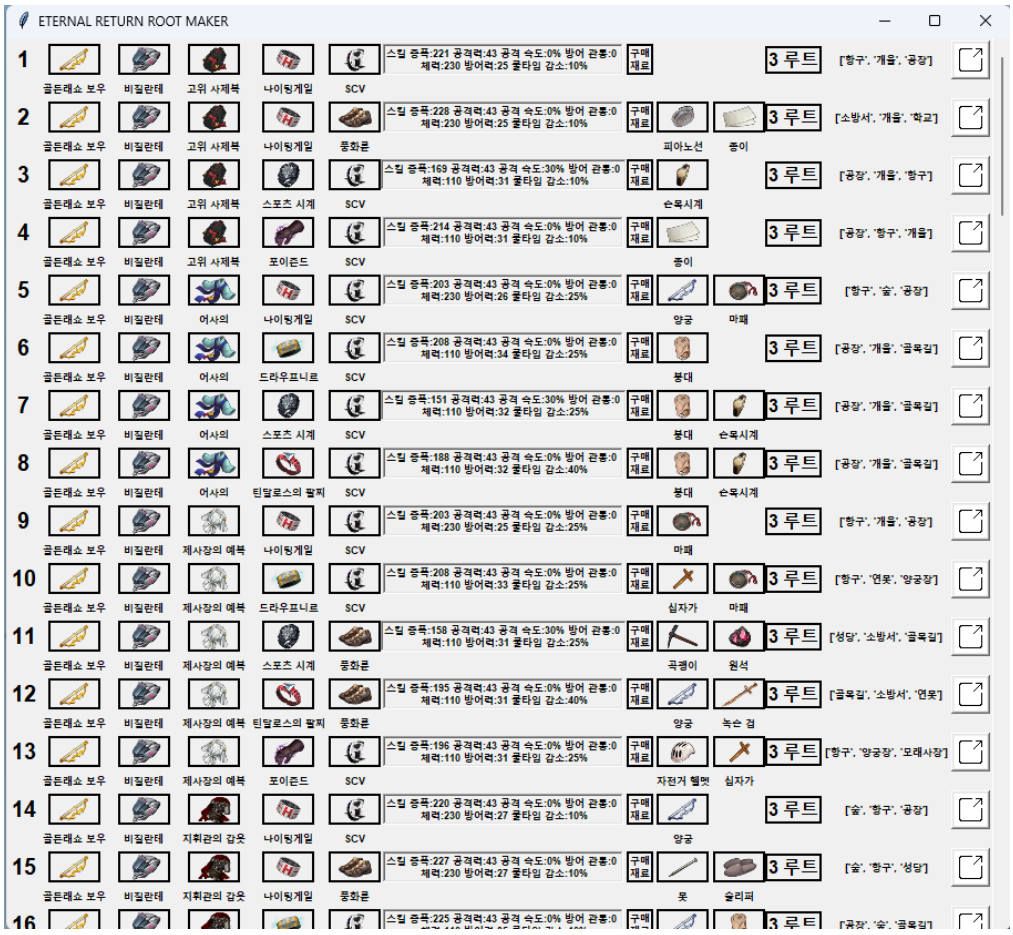
무기 종류 선택화면 : 캐릭터 이미지와 사용 가능한 무기 종류 표시 / 무기 종류를 선택하면 무기 선택창으로 이동



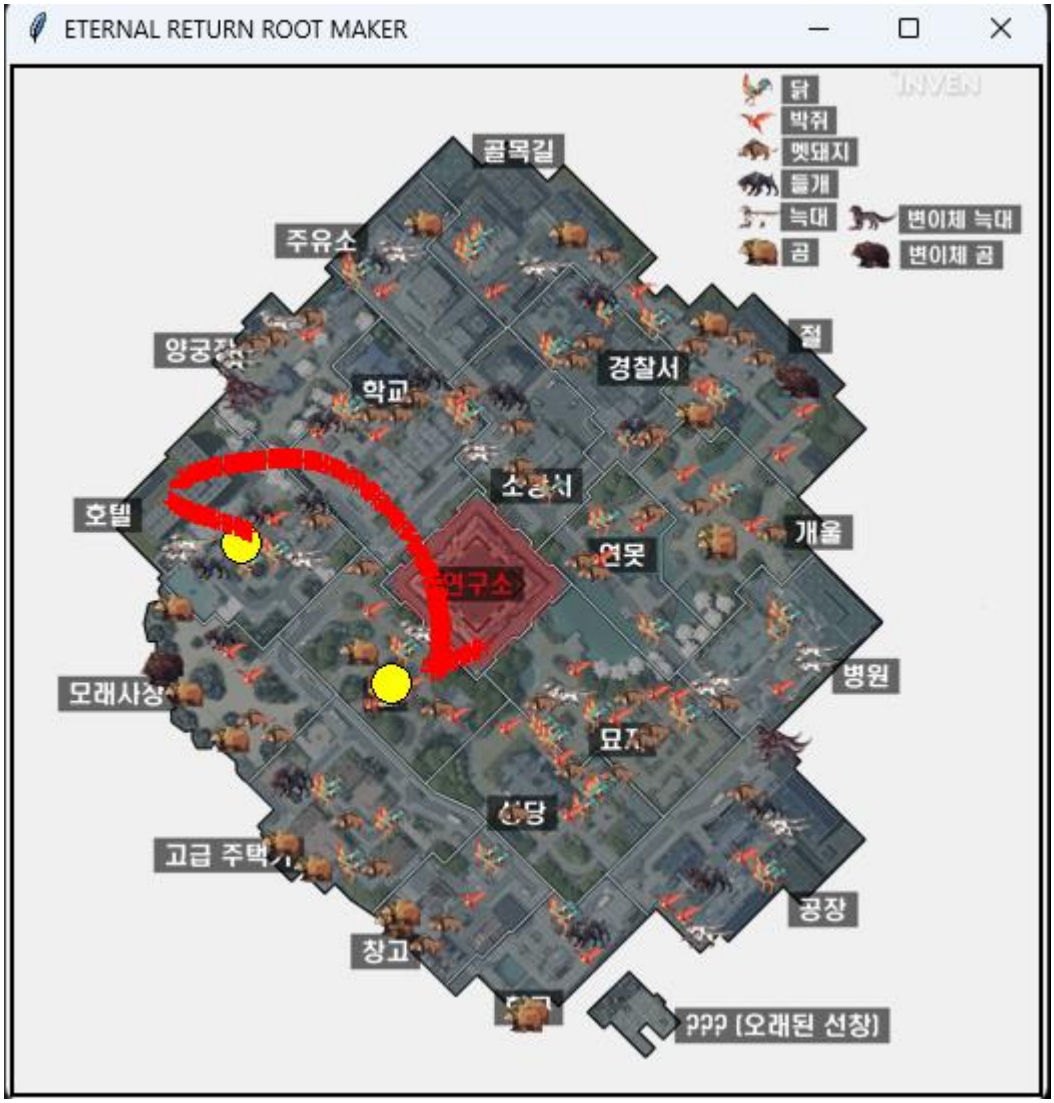
무기 선택화면 : 캐릭터 능력치 별 사용가능한 무기 이미지와 옵션 표시 / 무기를 선택하면 최대 구매 가능 재료 개수 선택창으로 이동



최대 구매 가능 재료 개수 선택 화면 : 개수를 선택하면 가능한 최적 루트 산출 후 루트 선택창으로 이동



루트 선택 화면 : 아이템별 옵션과 루트를 선택할 수 있음 / 바로가기 버튼을 통해 선택한 루트를 시각적으로 볼 수 있음'



지도 화면 : 지도에서 선택한 루트를 확인하고 그림 그리기 기능을 통해 직접 세세한 루트를 확인할 수 있게 함

프로젝트 후기 및 개선점

프로젝트를 진행하며 가장 크게 느낀점은 필요한 데이터를 추출하는 것 자체도 중요하지만 데이터를 가공하는 것에서 정보의 힘이 나온다는 것이었다. 정보들을 그냥 나열해 놓았을 때는 글자 덩어리였지만, 가공과 활용을 거치니 정보들이 모여 새로운 정보를 생성해 낸다는 것에서 정보의 힘을 느꼈다.

또한 이렇게 만든 정보를 GUI로 만드는 것이 생각보다 쉽지 않았다는 것을 느꼈다. 프로젝트를 진행하기 전에는 프론트엔드 즉 GUI는 핵심 기능 구현보다 난이도가 많이 떨어진다고 생각해였다. 하지만 프로젝트를 진행하며 GUI에서도 생각해야할 부분이 많고, 기능 구현 자체도 난이도가 있다는 것을 깨달았다. 또한 정보를 제대로 표현하지 못하면 아무리 잘 만든 프로그램이더라도 최고의 사용자 경험을 이끌어 낼 수 없다고 느꼈다.

마지막으로 프로젝트를 마친 후 코드를 다시 보며, 몇 가지 개선해야 할 부분을 발견했다. 객체지향적으로 프로그래밍을 한다면 코드를 좀 더 축약할 수 있다고 느꼈다. class 사용에 아직 많이 익숙하지 않아 잘 사용하지 못해 난잡한 코드가 나온 것 같아 class 사용에 익숙해진다면 더 좋은 프로그램을 만들 수 있을 것이라고 생각한다. 또한 아직 GUI가 완벽하지 않아 UI/UX 학습을 통해 조금 더 나은 사용자 경험을 제공하면 좋을 것이라 생각한다.