**REPO­­RT**

목차

1. 프로토타입 제목 : 길잡이(TrailBlazer)
2. 프로토타입 설명 : what / who / when / why / how
3. 사용한 자료구조
4. 사용한 알고리즘
5. Lessons Learned

로고, 텍스트, 상징, 등록 상표이(가) 표시된 사진

자동 생성된 설명

과목명  | 알고리즘

담당교수 | 조윤실

학과 | 컴퓨터공학과

학년 | 3

학번 | 201935266

이름 | 박영준

1. 프로토타입 제목

"Trailblazer"는 "trail"과 "blaze" 두 단어로 이루어진 이름입니다. "Trail"은 '길, 탐험로'를 의미하고, "blaze"는 원래 '불, 불길'을 뜻하지만, 나무에 표시를 남기는 행위도 포함됩니다. 옛날 탐험가들이나 숲속 길을 찾는 사람들이 나무에 흰 도장을 찍어 길을 표시했는데, 이를 "blaze"라고 불렀습니다. 따라서 "TrailBlazer"는 새로운 길을 열고, 그 길에 표시를 남겨 다른 사람들이 쉽게 따라올 수 있도록 하는 사람을 의미합니다.

“TrailBlazer”는 이러한 개념에 착안하여 학교 건물에 익숙하지 않은 복학생들이나 신입생들의 캠퍼스 라이프 경험을 향상시키는 것을 목표로 하고 있습니다.

2. 프로토타입 설명

What

"TrailBlazer"는 가천대학교 학생들을 위해 설계된 스마트 캠퍼스 내비게이션 어플리케이션입니다. 이 앱은 캠퍼스 내 건물과 시설에 대한 정확한 정보를 제공하며, 효율적인 길 찾기와 편의 시설 안내를 돕기 위해 개발되었습니다.

Who

이 어플리케이션은 특히 복학생이나 신입생, 그리고 캠퍼스 내 여러 건물의 위치와 경로에 익숙하지 않은 모든 학생들을 대상으로 합니다. 또한, 캠퍼스를 방문하는 외부인이나 교직원들에게도 유용하게 사용될 수 있습니다.

When

"TrailBlazer"는 가천대학교가 전국에서 13번째로 넓은 캠퍼스를 보유하고 있으며, 최근 건물 명칭 변경 및 새로운 건물의 추가로 인해 많은 학생들이 길을 찾는 데 어려움을 겪는 상황에서 개발되었습니다. 특히, 학기 시작과 같이 새로운 환경에 적응해야 하는 시기에 더욱 필요합니다.

Why

이 앱을 개발하게 된 계기는, 과거 복학 후 처음 학교에 돌아왔을 때 겪었던 혼란에서 비롯되었습니다. 당시 IT융합대학이 AI공학관으로 옮겨가며 강의가 끝나고 어디로 나가야 할 지 몰라 길을 잃었던 기억이 있었습니다. 또한, 네이버지도와 같은 기존 지도 서비스에서 무한대상이 2개로 표시되거나, 존재하지 않는 길이 버젓이 표시되는 등 여러 문제점들이 발견되었습니다. 이러한 문제들이 학우들에게 불편을 초래할 것이라고 생각되어, 보다 정확한 정보를 제공하고자 "TrailBlazer"를 개발하게 되었습니다.

How

"TrailBlazer"는 가천대학교의 최신 지도 데이터를 기반으로, 사용자가 입력한 위치 간의 최단 경로를 다익스트라 알고리즘을 통해 계산하여 제공합니다. 또한, 각 건물의 편의시설 정보와 운영시간 등을 포함하여, 학생들이 필요한 정보를 쉽게 찾을 수 있도록 설계되었습니다. 어플리케이션은 직관적인 사용자 인터페이스를 갖추고 있어, 간단한 입력만으로도 필요한 정보를 손쉽게 얻을 수 있습니다.

3. 사용한 자료구조

그래프 (Graph):

그래프는 NetworkX 라이브러리를 사용하여 구현되었으며, 노드와 엣지로 구성됩니다. 각 노드는 학교 건물을 나타내고, 엣지는 건물 간의 연결 경로를 나타냅니다. 그래프는 최단 경로 탐색과 같은 작업을 효율적으로 수행할 수 있도록 설계되었습니다.

딕셔너리 (Dictionary):

노드 속성:

각 노드는 딕셔너리 형태로 여러 속성을 가집니다. 예를 들어, 각 노드는 'label'과 'name' 속성을 가지며, 이를 통해 건물의 표시 이름과 실제 이름을 저장합니다. 이 딕셔너리는 graph.add\_node 메서드에서 실제 건물과 그 편의시설을 이중 딕셔너리로 지정하는 데 사용되었습니다.

편의시설과 이용 시간:

각 노드에는 'facilities' 속성이 포함된 딕셔너리가 있으며, 이 딕셔너리는 이중 딕셔너리 형태로 구성됩니다. 'facilities' 딕셔너리에는 각 편의시설의 이름을 키로, 위치와 운영시간 정보를 가지는 또 다른 딕셔너리가 값으로 저장됩니다. 이를 통해 각 건물의 편의시설 정보를 저장하고 탐색할 수 있습니다.

리스트 (List):

우선순위 큐:

heap\_dijkstra 함수에서는 우선순위 큐를 구현하기 위해 리스트를 사용합니다. heapq 모듈을 사용하여 이 리스트를 우선순위 큐로 사용합니다.

TrailBlazer에서는 그래프 구조를 사용하여 캠퍼스 내 건물과 경로를 관리하고, 딕셔너리를 사용하여 각 건물의 속성과 편의시설 정보를 저장합니다. 리스트는 우선순위 큐와 유사한 이름 검색에 사용됩니다. 이 모든 자료구조는 효율적인 탐색과 경로 계산을 가능하게 하여 사용자에게 정확한 정보를 제공합니다.

4. 사용한 알고리즘(3개 이상)

해시 테이블 (Hash Table):

해시 테이블은 딕셔너리 형태로 구현되어 있습니다. 딕셔너리는 키-값 쌍을 저장하고 빠르게 접근할 수 있는 자료구조로, 각 노드(건물)의 속성 및 편의시설 정보를 저장하는 데 사용됩니다.

사용 부분: 각 노드의 'name' 속성을 통해 건물의 정보를 빠르게 찾을 수 있습니다. 또한, 'facilities' 속성을 통해 건물의 편의시설과 운영시간을 효율적으로 관리합니다.

힙 트리 (Heap Tree):

힙 트리는 우선순위 큐로 구현되어 있으며, 다익스트라 알고리즘에서 최단 경로를 계산하는 데 사용됩니다. 힙 트리는 최소 힙을 사용하여 가장 짧은 거리의 노드를 효율적으로 추출할 수 있습니다.

다익스트라 알고리즘 (Dijkstra's Algorithm):

다익스트라 알고리즘은 그래프에서 한 노드에서 다른 노드까지의 최단 경로를 찾는 알고리즘입니다. 이 알고리즘은 모든 가능한 경로를 탐색하여 최단 경로를 계산합니다.

사용 부분: heap\_dijkstra 함수에서, 그래프 내의 두 노드 간 최단 거리를 계산하는 데 사용합니다.

유사도 매칭 (Similarity Matching):

유사도 매칭 알고리즘은 difflib 모듈을 사용하여, 사용자가 입력한 건물 이름과 가장 유사한 건물 이름을 찾아 힌트를 제공합니다.

사용 부분: InputForm 클래스의 update\_hint 메서드에서, 사용자 편의성을 위해 입력된 텍스트와 가장 유사한 건물 이름을 찾아 힌트를 제공하는 데 사용됩니다.

5. Lessons Learned : 결과물에 대한 자신의 피드백(PMI)

**P (Positive): 원코드로 멀티플랫폼 빌드가 가능하다는** Kivy 라이브러리를 제대로 배워보고 싶었습니다. 특히 레이아웃을 선언하는 부분이 AndroidStudio와 닮았던 부분이 마음에 들어https://www.youtube.com/watch?v=83C4tl8scoY와 Kivy 공식 문서를 참고해 학습 곡선대로 개발이 진행된 점은 긍정적이었습니다.

특히 사용자에게 인사 메시지를 띄우고, 종료에 대한 부분을 완벽하게 처리하는 등 사용자 경험 중심으로 개발이 진행되었던 점이 긍정적이였습니다.

**M (Minus):** 처음에는 다양한 기능을 추가하고 싶은 욕심에 관련 조사 없이 모듈을 무작정 만들었고, 그 결과 구조를 세 번 정도 바꾸게 되었습니다. 무엇보다 Kivy.garden과 Matplotlib이 서로 상호작용하지 않아(정확히는 3.2.0 버전 이후로 이벤트 처리 부분이 해결되지 않아 Threading을 사용하거나 Kivy.garden의 Agg 라이브러리를 사용해도) 원하는 결과물을 출력할 수 없었습니다. 이 점이 아쉬웠습니다.

처음에 원했던 결과물은 사용자의 위치를 받아와 가장 가까운 편의시설의 위치를 출력하는 알고리즘이었습니다. 그러나 관련 조사를 하지 않고 개발을 시작한 탓에 커스텀 맵을 만드는 작업에 많은 시간이 소요되었습니다. 결국, 커스텀 맵을 사용하려면 이미지 파일을 타일별로 나누고, 각각의 타일에 위도와 경도를 바인딩한 후, 사용자 위치와 비교해야 하는 매우 복잡한 작업임을 몰라 많은 시간을 허비했습니다. 이 과정에서 소프트웨어 공학에서 배우는 개발 모델이 이론적인 가이드라인이 아니라 개발 전 단계에서 지켜야 할 실용적인 가이드라인이라는 것을 배웠습니다.

특히, 원래 구조는 Kivy.garden의 MapView에 라벨링이 없는 학교 지도를 띄우고, 건물을 타일로 변환(gdal2)하는 작업이었습니다. 이 작업이 생각만큼 간단하지 않았습니다. 우선 gdal 모듈을 설치하는 것부터가 난관이었고, 설치 후에도 자동 변환이 아닌 건물 하나하나마다(타일 하나마다) 위도와 경도를 바인딩해야 했습니다. GPS로 받아온 위도, 경도와 비교해 거리나 경로를 출력하려 했지만, 중간에 너무 많은 오류가 발생해 결국 포기하게 되었고, 이로 인해 시간을 많이 낭비했습니다.

**I (Interesting):** 그럼에도 불구하고 세 번의 구조 변경을 통해 확장성과 유지보수성이 매우 유망한 어플리케이션이 되었습니다. 나중에 시간이 나면 건물 사이사이의 가중치를 스마트폰 어플리케이션으로 측정해 정확한 가중치와 편의시설 목록을 업데이트하면 사용자 경험이 크게 향상될 것입니다. 또한 Kivy의 멀티플랫폼 어플리케이션 특성 상, 현재 코드에서 화면 크기만 RelativeLayout으로 수정하고 위젯을 적절히 배치하면 동일한 코드로 빌드가 가능했습니다.

마지막으로 처음에 고생을 많이 한 부분인데, Kivy는 비 라틴계 문자에 대해 모든 폰트를 지정해줘야 출력이 가능합니다. 특히 토스트 메시지의 제목에는 font\_name 속성을 사용할 수 없었습니다. 이에 생성형 AI에게 문의했더니 custom title 바를 추가해 제목을 덮으라고 하거나(llama3), 그냥 타이틀에 적으면 출력된다고 우기거나(gpt4o) 했습니다. 그러나 실제 정답은 에러 메시지에 있었습니다. 에러 메시지를 잘 읽어보면 TypeError: Properties ['font\_name'] passed to **init** may not be existing property names. Valid properties are ['title\_font']라 나와 있었고, font\_name 대신 title\_font 속성을 사용하니 한글이 정상적으로 출력되었습니다.

이를 통해 생성형 AI에 지나치게 의존하지 말아야겠다는 생각을 하게 되었습니다.

이 과정에서 배운 교훈들은 향후 프로젝트에서 더욱 신중하게 접근하고, 계획을 세우는 데 큰 도움이 될 것입니다.