지난 시간 발표한 ppt 복습.

프로젝트 주제는 시위대 행진 경로를 Dynamic Graph로 표현하기.

프로젝트 목표는 시간에 따른 행진 Dynamic Graph의 변화에 맞추어

차벽(바리케이드) 예상 교차로, 주요 경력 배치 교차로 등을 예측(프로젝트 진행하면서 더 추가될 예정).

저번에도 말했듯이 최종 프로젝트 결과는 왼쪽 사진 자료와 비슷하게 보여질 예정.

오른쪽 설명과 같이 Node는 교차로, Edge는 교차로 사이의 도로, Weight는 도로의 길이, Orientation은 행진의 진행 방향.

Dynamic한 부분은 시간이 지남에 따라 행진 경로상의 Node들이 추가/삭제 되는것.

이번 Problem Solving에서 설명드릴 목차.

간단하게 하나씩 설명 드리자면 만들 소프트웨어 종류는 PC 프로그램.

언어는 Java를 쓸 것이고

아까 사진 자료와 같이 보여드린 화면은 Java의 GUI를 이용하고 제가 따로 찾은 깨끗한 지도 위에서 모든것이 진행될것.

Dynamic Graph를 어떻게 화면에 표시할지는 아직 고민중인데, Jpanel를 이용해서 그리거나 픽토그램 등을 이용해서 표시할 예정.

프로그램 조작법은 여러가지 버튼들이 화면 상단에 있을건데 주로 이 Dynamic Graph에서 가장 중요한 요소 중 하나인 시간은 조작하는 버튼들과 UI들로 구성될 예정.

자료를 프로그램에 입력하는 방법은 엑셀파일을 이용하여 행진의 경로와 여러 정보를 정해진 양식에 맞추어 작성하면

그에 대한 정보를 프로그램에서 알맞게 추출.

Dynamic Graph의 자료 구조는 HashMap과 ArrayList를 이용할 예정. HashMap 안에 ArrayList가 있는 방식. 추후에 설명.

프로그램 디자인 패턴은 MVC패턴을 이용할 예정.

이번 슬라이드에서는 전 슬라이드에서 말씀드린 디스플레이 부분을 세세하게 설명해드림.

현재 화면에 보이시는게 실제 프로그램 디스플레이 화면이 될 것.

하나씩 설명드리자면 지도는 네이버 지도에서 스크린샷을 통해서 원하는 부분만 추출했고

보이는 동그라미들이 Node, 그리고 화살표가 Edge임.

파란색 노드들은 가상의 행진 1이 현재 지나가고 있는 교차로의 노드들,

주황색 노드들도 가상의 행진 2가 현재 지나가고 있는 교차로의 노드들.

회색 노드들은 Inactive 노드들이라고 현재 행진이 지나가고 있지 않은 노드들.

이 노드들은 행진이 지나가게되면 Active로 바뀌어 색깔이 생기고 엣지 화살표도 생김.

결과적으로는 해당 지도상의 모든 교차로를 일단 Graph에 저장해두고 행진이 지나가는 부분만 Active로 해놔서 보이게 표시하는셈.

빨간색과 초록색 노드들은 가상의 차벽(바리케이드) 예상 교차로랑, 주요 경력 배치 예상 교차로. 이것들도 시간이 지남에 따라 변화.

마지막으로 상단의 UI는 현재 행진 관련 시간이 표시되고 오른쪽에는 설명과 같이 시간 조작이 가능한 여러 버튼들로 구성되어있음.

이번 슬라이드에서는 Data Set이 엑셀 파일에서 어떻게 Input되는지 간단히 설명해드림.

왼쪽의 사진은 전 슬라이드에서 말씀드린 지도상의 전체 주요 교차로 Data Set 양식.

교차로와 인접 교차로상의 거리 정보를 가지고 있음. 거리 정보를 자동으로 처리해주는 방법 몇 개 찾았으나

길찾기 방식이어서 교차로 이름이 없는 곳이거나 경찰 내부에서 사용하는 교차로들이 검색이 안되는 경우가 많아

그냥 제가 측정하는게 더욱 확실하고 정확할 것으로 판단되어 하나씩 네이버 지도 직선거리 측정을 이용해서 작성.

오른쪽 아래 사진처럼 한 교차로에서 인접 교차로 까지의 거리를 마우스를 광클릭하여 측정. 총 250번 정도 했음…

정말 힘들었음…..

오른쪽 위 사진은 실제 행진에 대한 정보 Data 양식. 저번주에 보여드린 ppt에서 서울청에서 직접 pdf로 다운받을 수 있는

정보를 위 사진과 같이 엑셀 양식을 만들어 간편하게 입력할 수 있도록 작성.

다음 슬라이드로 넘어감.

이번 슬라이드에서는 선택한 Data Structure와 왜 선택했는지에 대해 설명해드림.

일단 Adjacency Matrix 대신 Adjacency List를 선택한 이유는, 다들 구글링 해봐서 알겠지만

공간 복잡도가 Matrix는 Node의 개수 제곱인데 반해 List는 Node의 개수 + Edge의 개수.

제 프로젝트에는 총 Node가 66개 있으니까 당연히 List를 사용하는게 더 메모리를 절약할수 있음.

그리고 Adjacency List 안에서의 자료구조는 사진과 같이 HashMap 안에 ArrayList가 있는것. 아직 ArrayList와 Linked List사이 고민중이라서 둘다 표\시.

HashMap을 선택한 이유는 Adjacency List를 사용하면 각 원소에 대한 접근성이 떨어지는데 이를 최대한 살리려

접근성이 뛰어난 HashMap을 선택. 순차적으로 원소들을 검색하지 않아 시간 비약적으로 절약.

Map의 특징인 Key로 Value를 얻어내는 점이 가장 큰 장점.

그리고 각 원소들의 Value들은 ArrayList 또는 Linked List일 예정인데 둘다 장단점이 팽팽해서 아직 못고름.

이게 마지막 슬라이드고 여기서는 보시다시피 UML Diagram이 있음.

아까 말씀드린 MVC 패턴을 위 사진처럼 사용할 예정.

Edge, Node, Graph를 Model로 그리고 View와 Controller가 있는 방식.

정말 무슨 Class가 있고 그 안에 뭐가 있을 예정이다만 표시한것.

각 Class들간 관계와 메서드들도 마찬가지.

이것을 토대로 프로젝트 완성 여부를 다음 발표때도 알려드림.