⟨SE-Team⟩

⟨ADD-ON⟩

설계 산출물

문서버전	3.0
문서 ID	SE-2019-002
최종 변경일	2019-12-04
문서상태	릴리즈

요 약

SIM 와 함께 사용하는 ADD-ON 시스템의 설계 산출물을 기술

서브 시스템의 구성과 각 서브 시스템의 구조를 기술.

주요 산출물

- 아키텍처도
- 클래스도
- 교류도
- 상태도

컴퓨터과학부 2015920062 홍성현

컴퓨터과학부 2016920027 오준영

표 1. 문서 변경 기록

문서이름		〈SE-Team〉〈ADD-ON〉 설계 산출물	
문서 ID		SE-2019-002	
버전		변경일	설명
1	1	2019-11-02	아키텍쳐도, 메인 클래스도, 교류도 초안을 작성했다.
	2	2019-11-03	객체 Show_result 의 상태도를 작성했다. Use Case ‘show map data,’ ‘show menu’의 교류도를 다음과 같이 수정했다. - Show menu 에 있던 맵 정보 보여주기 관련 메소드들을 ‘show map data’ Use Cas 의 교류도로 옮겼다.
	3	2019-11-04	Use Case ‘Show result’의 교류도를 다음과 같이 수정했다. -14 번 메시지 Show_message(‘robot found hazard spot!’)을 추가했다.
	4	2019-11-05	객체 map_data 의 상태도를 작성했다. 객체 Show_result 의 상태도를 작성했다.
2	0	2019-11-06	중간 릴리즈
	1	2019-11-12	SaveData, ShowMapData 클래스를 추가했다. MapData 클래스를 정적 클래스로 바꾸었다. 이를 통해 여러 클래스에서 MapData 클래스의 저장 내용을 공유할 수 있다. ShowMenu 클래스에 변수 sd, smd 를 추가했다. 각각의 변수들은 SaveData, ShowMapData 클래스의 인스턴스를 저장한다. 위의 내용들을 클래스 다이어그램에 반영했다.
	2	2019-11-13	클래스도를 수정했다. - SaveData 클래스를 업데이트했다. showMessage 메소드 제거 - ShowResult 클래스를 만들었다.
	3	2019-11-15	클래스도를 수정했다. - ShowResult 클래스를 업데이트했다. drawPath, removePath 메소드 추가
	4	2019-11-16	클래스도를 수정했다. - ShowResult 클래스에 showRobotMovement 메소드 추가

2	5	2019-11-17	클래스도를 수정했다. - MessageController 클래스 추가 showMessage 메소드 생성 -〉 메시지 박스 생성 담당 - ShowResult 클래스 수정 generateRandomSpot 메소드 추가 changePath 메소드 추가 - SavaData 클래스 수정 mapDataSave 메소드의 이름을 saveInputData 로 바꿨다.
	6	2019-11-20	유즈케이스도에 변화가 있어 그에 맞게 교류도를 업데이트했다. 클래스도를 수정했다. 새로운 클래스 ControlRobot 을 만들고, 여기에 경로 생성 및 반환에 관련된 메소드들을 넣었다.
	7	2019-11-24	클래스도를 수정했다. 새로운 클래스 ControlPath 를 만들고, ControlRobot 에서 경로와 관련된 기능들을 새 클래스로 옮겼다. 로봇의 기능을 구현하는 SIM 클래스를 작성했다.
	8	2019-11-27	SIM 클래스 내부에 로봇의 오동작을 구현하고, ShowResult 클래스 내부에 맵 외곽을 그려주는 메소드를 추가했다. 이를 클래스 다이어그램에 반영했다.
	9	2019-11-30	클래스 이름 변경 ControlPath -〉 PathController ControlRobot -〉 RobotController ShowResult -〉 ShowRobotMovement 클래스 통합 MessageController 클래스를 WindowDesign 클래스와 통합
3	0	2019-12-04	릴리즈

1. 개　　요

1.1 목　　적

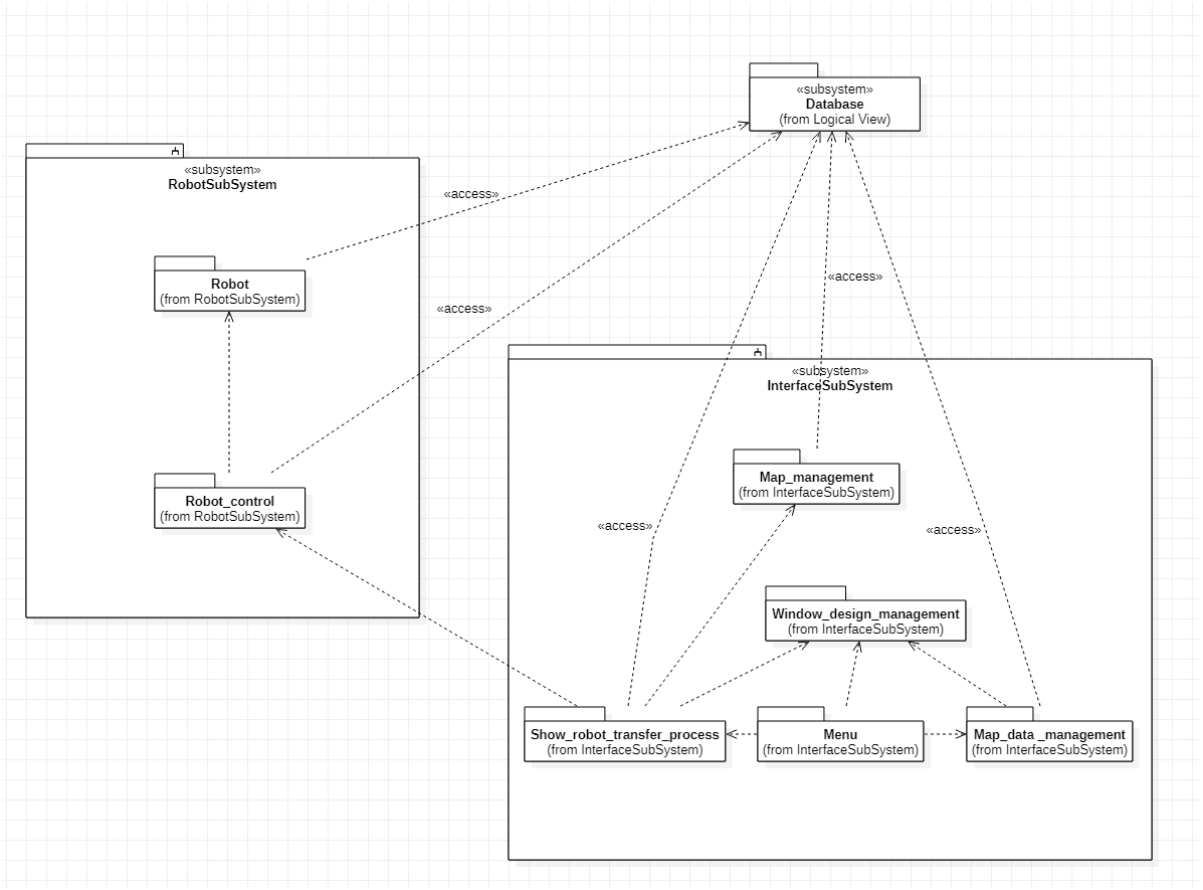
본 문서는 “Mobile Robot Controller 와 함께 사용하는 ADD-ON” 의 설계 산출물을 기술한다.

- 전체 시스템이 어떤 서브 시스템으로 구성되어 있는지 “아키텍처도”를 통해 보여준다.
- 전체 시스템이 어떤 클래스를 가지고 있고, 각 클래스들 간에 어떤 관계가 있는지를 “클래스도”를 통해서 보여준다.
- 각 Use Case 별로 객체 간에 발생하는 동적인 행위를 “교류도”를 통해서 보여준다.
- 두 객체의 상태 변화를 “상태전이도”를 통해서 보여준다.

1.2 참고 문헌

없음.

2. 아키텍처도



전체 시스템은 아래와 같은 서브 시스템으로 이루어진다.

- Robot 서브시스템
- Database 서브시스템
- Interface 서브시스템

Robot 서브시스템

- mobile robot 의 세 가지 센서 (Positioning sensor, Hazard sensor, Color blob sensor)를 구현하고 있으며, 이들 센서에서 오는 정보를 바탕으로 mobile robot 에게 동작을 지시하는 역할을 한다.
- python 언어를 사용하여 작성된다.

Database 서브시스템

- Map 의 크기와 로봇의 시작 지점, 목표 지점, 알려져 있는 hazard spot 좌표를 MapData 로, 그리고 만들어진 경로를 PathInfo 로 저장하는 역할을 한다.
- python 언어를 이용하여 작성된다.

Interface 서브시스템

- Operator 에게 메뉴를 띄워주고, Operator 가 고른 메뉴에 따라 map data 저장, 저장된 map data 출력, 로봇의 이동 과정 출력과 같은 기능들을 제공해주는 역할을 한다.
- python 언어의 라이브러리인 matplotlib, PyQt5 를 이용한다.

3. 서브시스템 세부 설계

3.1 Robot 서브시스템

Robot 서브시스템은 mobile robot 의 세 가지 센서들을 통해 얻은 정보를 바탕으로 mobile robot 의 동작을 지시하는 역할을 한다. Mobile robot 을 위한 경로를 만드는 기능을 수행하기도 한다.

- map 정보는 Database 서브시스템을 통해서 얻는다.
- Robot 서브시스템의 Robot_control 패키지는 Interface 서브시스템에서 사용된다.

Robot 서브시스템은 아래와 같은 함수들로 구성되어 있다.

Robot 패키지

- 인터페이스 IRobotInterface

  로봇의 동작 및 hazard sensor, color blob sensor, positioning sensor 에 관한 인터페이스이다. 클래스 SIM 이 구현한다.

- 클래스 SIM

이 클래스는 로봇의 동작 매커니즘을 담고 있다. RobotController 클래스에 의해 컨트롤된다.

- __init__(self, dir) : 현재 저장되어 있는 경로 정보를 토대로 구한 로봇의 방향 dir 을 direction 에 넣어 초기 방향으로 설정한다. self.position 에는 mapData 에 저장되어 있는 로봇의 시작위치를 넣고, 로봇이 발견한 hidden hazard spot 들을 저장하는 빈 리스트 foundHazardSpot 을 선언한다.

- _checkError(self, checkSpot) : 바로 앞이나 두 칸 앞에 hazard spot 이 있을 때(hidden hazard spot 도 포함), 또는 앞으로 두 칸 움직이면 맵 밖으로 나가버리는 경우 앞으로 두 칸 움직이는 오동작을 배제하기 위해 사용한다. 이 메소드는 SIM 클래스 내부에서만 사용된다.

- moveForward(self) : 로봇을 앞으로 한 칸 이동시킨다. 로봇이 움직이지 않거나 앞으로 두 칸 움직일 가능성이 있다.

- turnCW(self) : 로봇을 시계방향으로 90 도 회전시킨다. direction 값에 1 을 더한 뒤 4 의 나머지 연산을 수행한다.

  direction 0 : 위 1 : 오른쪽 2 : 아래 3 : 왼쪽

- positioningSensor(self) : 현재 로봇의 위치와 방향을 리턴한다.

- colorBlobSensor(self) : 인근 지점에 color blob spot 이 있는지 확인하고 있을 경우 해당 지점들의 방향을 리스트에 넣어서 반환한다.

- hazardSensor(self) : 현재 로봇의 전방 한 칸 앞에서 위험 지역을 찾고 발견했을 시 foundHazardSpot 배열에 추가하고 True 를 리턴한다. 없을 경우 False 를 리턴한다.

- getFoundHazardSpot(self) : 지금까지 로봇이 발견한 hidden hazard spot 들의 리스트를 반환한다.

Robot_control 패키지

- 인터페이스 IRobotCtrl

  클래스 RobotController 가 구현한다. 로봇을 회전시키거나 로봇의 센서 정보를 받아오거나 로봇의 현재 방향 체크, 로봇 동작 지시와 관련되어 있는 인터페이스이다.

- 클래스 RobotController

  이 클래스는 로봇의 동작을 컨트롤한다. ShowRobotMovement 클래스에서 이 클래스의 인스턴스가 생성된다.

  - __init__(self, direction, pathController, mapSize) : ShowRobotMovement 클래스로부터 로봇의 초기 방향과 PathController 클래스의 인스턴스를 매개변수로 받는다. 로봇의 초기 방향은 ShowRobotMovement 에서 pathController 를 통해 생성한 초기 경로에 따라 결정된다. 그리고 SIM 클래스의 인스턴스를 생성한다. 경로를 저장할 리스트를 선언하고 맵의 크기, 경로가 저장된 리스트 상에서 현재 로봇이 있는 좌표의 인덱스를 생성한다. 또한 다음 지점과 현재 지점 간의 좌표 차이를 통해 로봇이 봐야하는 방향을 결정하게 해주는 pointToDirection 딕셔너리를 생성한다. pointToDirection 딕셔너리는 다음과 같은 구조로 되어있다.

    {(0, 1) : 0, (1, 0) : 1, (0, -1) : 2, (-1, 0) : 3}

  - turnCW(self) : SIM 의 turnCW 메소드를 호출하여 로봇을 시계방향으로 90 도 돌린다.

  - getSensorInfo(self) : SIM 의 color blob sensor, hazard sensor 를 실행시키고 결과 리스트의 길이가 1 이상이면 리스트의 원소들을 참고하여 color blob spot 좌표를 계산해 리스트에 저장한다.

    만일 로봇이 현재 경로 상의 다음 지점을 향하고 있다면 hazard sensor 값을 받아온다. hazard sensor 를 통해 hidden hazard spot 이 검출되면 현재 로봇의 위치, 방향을 고려하여 hidden hazard spot 의 좌표를 계산하여 저장해 둔다.

최종적으로 현재 로봇 위치 전후좌우에 존재하는 color blob spot 의 좌표 리스트와 함께 로봇 전방에 hazard spot 이 존재한다면 해당 좌표를, 존재하지 않으면 (-1, -1)을 리턴한다.

- _calCBCoordinates(self, cbDirection, curPosition) : cbDirection 안의 원소를 참조하여 hidden color blob spot 좌표를 계산하여 리스트에 저장한다. 계산이 다 끝나면 저장한 리스트를 리턴한다. 이 메소드는 RobotController 클래스 내부에서만 사용된다

- checkDirection(self) : 현재 로봇이 경로상의 다음 가야할 지점을 향하고 있는지 확인한다. 방향이 맞지 않으면 turnCW() 메소드를 통해서 로봇의 방향을 바꿔준다.

- commandMovement(self) : checkDirection() 메소드를 이용해서 로봇이 경로 상의 다음 지점을 보고 있는지 확인한다. checkDirection 값이 False 라면 다른 곳을 보고 있다는 뜻이므로 turnCW() 메소드를 통해 로봇의 방향을 바꿔준다. True 라면 로봇을 앞으로 한 칸 이동시킨다. 로봇의 positioning sensor 를 다시 한 번 호출하여 그 값을 리턴한다.

- incPathNum(self) : pathNum 을 1 만큼 올려준다.

- getFoundHazardSpot(self) : SIM 클래스의 getFoundHazardSpot() 메소드를 통해 지금까지 로봇이 발견한 hidden hazard spot 을 리턴한다.

- 인터페이스 IPathCtrl

  - PathController 에서 구현한다. 이 인터페이스의 구현을 통해 새로운 경로를 만들거나 저장되어 있는 경로를 반환할 수 있다.

- 클래스 PathController

  이 클래스는 경로 정보를 담고있는 클래스 PathInfo 를 컨트롤한다.

  - __init__(self, dir=-1) : PathInfo 클래스의 인스턴스를 생성한다. MapData 클래스에서 맵 크기 정보를 받아온다.

- getPath(self) : 경로정보를 반환한다. RobotController, ShowRobotMovement 클래스에서 사용된다.

- createPath(self, start, foundHazardSpot=[]) : 현재 로봇의 위치에서 시작하여 모든 목표지점으로 향하는 경로를 만든다. 현재 맵 상에 진하게 표시된 hazard spot 들은 피하는 경로를 만들어야 한다. 경로를 만드는 알고리즘으로는 a* 알고리즘을 사용한다. 메소드 수행 과정은 아래와 같다.

  1. 현재 로봇과 가장 가까이 있고, 아직 발견하지 않은 목표 지점을 찾는다(Manhattan distance 이용). 이 지점을 첫 번째 도착 지점으로 설정한다.

  2. 현재 로봇이 있는 지점을 출발 지점으로 하여 도착 지점까지의 경로를 a* 알고리즘을 통해 구한다.

  3. 구한 경로를 클래스 변수 path 에 추가한다.

  4. 출발 지점을 목표로 삼았던 도착 지점으로 바꾸고, 이 지점에서 가장 가깝고 아직 방문하지 않은 다른 목표 지점을 찾아 그 지점을 도착 지점으로 하여 a* 알고리즘을 수행한다.

  5. 모든 목표지점까지의 경로가 만들어질 때까지 반복

- _aStar(self, start, end) : start 지점에서 end 지점까지 가는 경로를 구하는 a* 알고리즘을 수행한다. 세 개의 2 차원 리스트 close list, open list, parent 를 관리한다. close list 는 이미 방문했거나 hazard spot 인 지점을 넣는다. open list 는 현재 지점 상하좌우 지점에서 close list 에 들어가지 않는 지점들을 넣는다. parent 는 특정한 후보 지점을 경로에 넣을 때 후보 지점의 부모를 현재 지점으로 설정할 때 사용한다. 목표 지점에 도착한 뒤 parent 리스트를 통해서 목표 지점에서 출발 지점까지의 경로를 구하고, 이를 역순으로 배치해서 최종 경로를 구한다.

- _cal(self, start, end) : a* 알고리즘에서 사용하는 메소드이다. 먼저 start 를 openList 에서 closeList 로 옮긴다. 인접한 지점 중 closeList 에 속하지 않고 hazard spot 이 아닌 점을 openList 에 추가하고 현재 지점을 부모로 지정한다. 이 과정이 끝나면 openList 에 있는 점들의 f 값을 구해서 이를 우선순위 큐에 넣는다. 우선순위 큐를 통해 f 값이 최소인 점을 뽑아 이 점을 openList 에서 closeList 로 옮기고 자기 자신(_cal)을 재귀호출한다. start 에는 우선순위 큐에서 뽑은 점을, end 에는 기존 end 값을 넣는다.

- _g(self, oldSpot,candidateSpot) : _cal 메소드에서 사용한다. 경로 출발 지점에서 후보 지점(candidateSpot)까지의 실제 이동 거리(g)를 구하여 2 차원 리스트 arrayG 에 저장하는 메소드이다. arrayG 에 저장되어 있는 이전 좌표(oldSpot)의 g 값에 1 을 더하여 후보 좌표의 g 값을 구한다.

- _isHazardSpot(Self, x, y) : _cal 메소드에서 사용한다. 지점 (x, y)가 hazard spot 인지를 판정한다.

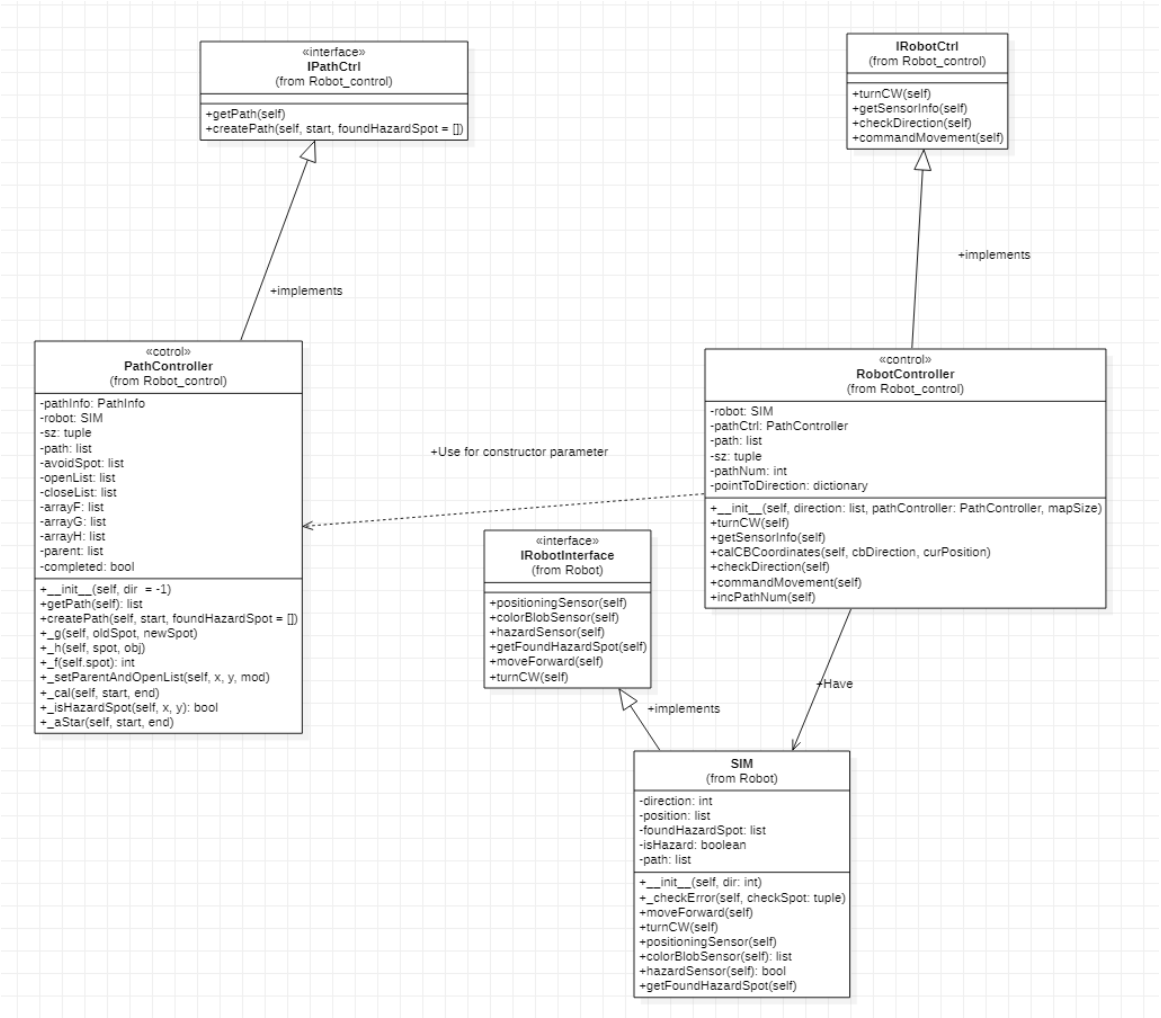
- _h(self, candidateSpot, objSpot) : _cal 메소드에서 사용한다. 현재 위치 currentSpot 에서 objectSpot 까지의 대략적인 거리를 manhattan distance 를 구하여 리스트 arrayH 에 저장한다. 이 값은 hazard spot 을 무시하고 구한 heuristic 값이다.

- _f(self, candidateSpot) : 후보 지점의 g 값과 h 값을 더하여 f 값을 구한다.

- _setParentAndOpenList(self, x, y, mod) : 좌표 (x, + mod[0], y + mod[1]) (후보 지점)을 확인한다. 해당 지점이 openList 안에 있을 경우 (x, y)(현재 지점)의 g 값 + 1 과 후보 지점의 g 값을 비교한다. 만일 현재 지점의 g 값 + 1 이 더 작다면 이는 기존에 후보 지점까지 가는 경로보다 현재 지점을 거쳐서 후보 지점까지 가는 비용이 더 작다는 의미이므로 후보 지점의 부모를 현재 지점으로 바꿔준다.

- _isHazardSpot(self, x, y) : 좌표 (x, y)에 맵 정보로 알려져 있던 hazard spot 이나 로봇이 찾아낸 hidden hazard spot 이 있는지 여부를 True/False 로 반환한다. _cal 메소드 안에서만 사용한다.

3.1.1 메인 클래스도



RobotController

- Database 에서 맵 정보를 얻어와 mobile robot 을 위한 경로를 만들고, 이를 다시 Database 의 PathInfo 에 저장한다. Interface Sub System 의 ShowRobotMovement 클래스에서 사용한다.
- 경로를 보고 mobile robot 에게 적절한 동작을 지시한다.
- mobile robot 의 센서를 통해 hazard spot, color blob spot 정보를 얻는다.

SIM

- RobotController 의 명령을 받고 앞으로 1 칸 이동이나 시계 방향으로 90 도 회전 중 하나의 동작을 수행한다.
- 세 개의 센서를 이용하여 Robot controller 현재의 mobile robot 좌표, 보고있는 방향, 전방 1 칸에 hazard spot 존재 여부, 전후좌우 1 칸에 color blob spot 존재 여부를 알려준다.

PathController

- Database 로부터 받은 맵 정보를 바탕으로 경로를 만든다. 경로를 만들 땐 A* 알고리즘을 사용한다.

3.2 Database 서브시스템

Database 서브시스템은 Operator 에게서 Map 크기, mobile robot 시작 지점, hazard spot 좌표, object spot 좌표를 입력 받아 이를 Map Data 로 저장하는 역할을 수행한다. 그리고 Interface 서브시스템으로부터 경로 정보를 받아 저장하는 역할도 수행한다. Interface 서브시스템에서 생성한 hidden hazard spot 좌표, hidden color blob 좌표를 입력 받아 이를 관리하는 역할도 수행한다.

- Map data 는 Interface 서브시스템에서 받고, Path Info 는 Robot 서브 시스템의 PathController 에서 받는다..

- 다양한 get() 메소드를 통해서 저장되어 있는 정보를 Interface 서브시스템, Robot 서브시스템에 제공한다.

Database 서브시스템은 아래와 같은 클래스들로 구성되어 있다.

- 인터페이스 IMapInfo

  저장되어 있는 맵 데이터를 반환하고, hidden spot 들을 설정하며 로봇이 찾은 hidden spot 들을 리스트에서 제거하는 등의 역할과 관련된 인터페이스이다. MapData 클래스가 구현한다.

- 클래스 MapData

  이 클래스는 정적 클래스이다.

  - __init__(self, mapSize, startSpot, objectSpot, hazardSpot) : 매개변수로 받은 맵 정보들을 저장해둔다. objectSpot 과 hazardSpot 은 투플을 원소로 가지는 리스트형으로 받는다.

  - setHiddenData(hazard, cb) : 매개변수로 받은 숨겨진 hazard spot 들과 color blob spot 들을 각각 리스트 hazardSpotH, colorBlobH 에 저장해둔다.

  - getHazardH() : 리스트 hazardSpotH 를 리턴한다.

  - getCbH() : 리스트 colorBlobH 를 리턴한다.

  - getMapSize() : 맵 크기를 리턴한다.

  - getStartSpot() : 시작 지점을 리턴한다.

  - getBackObjectSpot(list) : objectSpot 리스트를 처음 입력했던 상태로 돌려놓는다.

  - getObjectSpot() : 목표 지점들을 리스트의 형태로 리턴한다.

  - getHazardSpot() : hazard spot 들을 리스트의 형태로 리턴한다.

  - removeHiddenHSpot(point) : 다른 지역에서 동일한 hidden hazard spot 이 다시 검출될 수 있고, 오류가 발생하는 것을 방지하기위해 발견한 hidde hazard spot 을 삭제한다.

  - removeHiddenCBSpot(point) : 다른 지역에서 동일한 hidden color blob spot 이 다시 검출될 수 있고, 오류가 발생하는 것을 방지하기 위해 발견한 hidden color blob spot 을 삭제한다.

- 인터페이스 IPathInfo

  PathInfo 클래스가 구현한다. 이 인터페이스의 구현을 통해 경로 정보를 저장하고, 저장되어 있는 경로 정보를 반환할 수 있다.

- 클래스 PathInfo

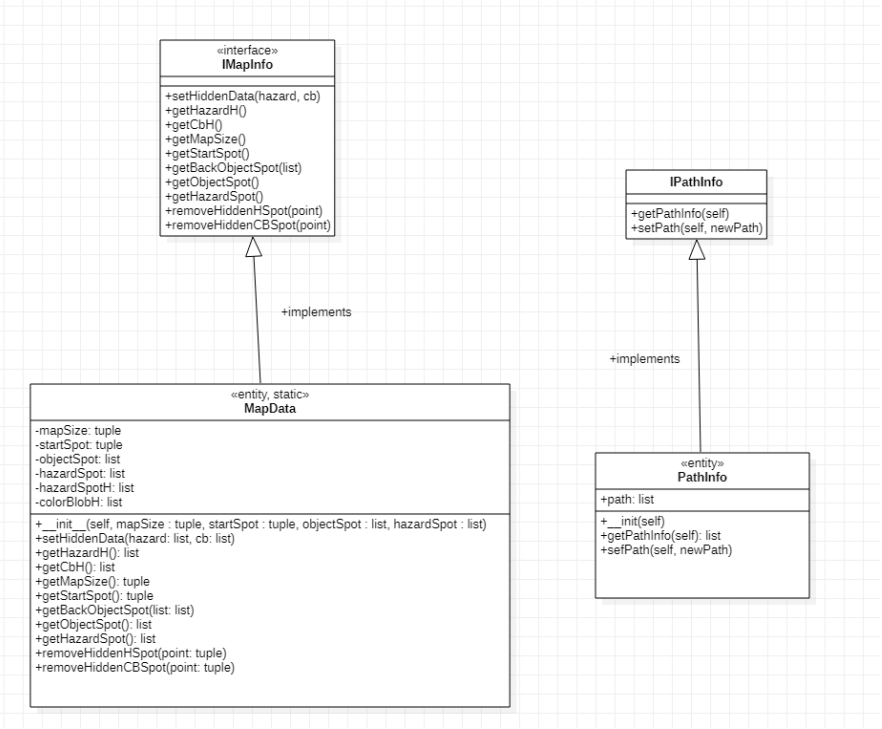
  이 클래스는 경로 정보를 저장하고 있다. PathController 가 이 클래스의 정보를 관리한다.

  - __init__(self) : 비어있는 리스트 path 를 선언한다.

  - getPathInfo(self) : 현재 저장되어 있는 경로 정보를 반환한다. 경로는 좌표 리스트로 이루어져 있다.

  - setPath(self, newPath) : 매개변수로 받은 newPath 를 클래스 변수 path 에 저장한다.

3.2.1 메인 클래스도



MapData

- 인스턴스 생성 시에 다양한 map 정보들을 받고 이를 저장한다.
- 다양한 get() 메소드들을 통해서 저장한 정보들을 Interface 서브시스템, Robot 서브시스템에 제공한다.
- setHiddenData() 메소드는 Interface 서브시스템에서 사용되며, 알려지지 않은 hazard spot 및 color blob spot 좌표를 저장한다.
- removeHiddenHspot(), removeHiddenCBSpot() 메소드는 mobile robot 이 발견한 hidden hazard spot 과 hidden color blob spot 을 리스트 hazardSpotH 에서 제거하는 기능을 수행한다.

3.3 Interface 서브시스템

Interface 서브시스템은 프로그램을 처음 실행했을 때 operator 에게 메뉴를 제공해주는 기능을

수행한다. 그리고 operator 에게 map, mobile robot 의 움직임, mobile robot 이 발견하는 hazard spot 과 color blob spot 등을 시각적으로 보여주는 역할을 한다.

- Interface 서브시스템은 save data, show robot movement, show map data, quit 의 4 가지 메뉴를 제공한다.

  save data 는 map data 를 Database 서브시스템에 저장하는 기능을 수행한다. show map data 는 MapData 서브시스템에 저장되어 있는 데이터를 보여준다. Show robot movement 는 SIM 서브시스템을 통해서 mobile robot 의 동작을 지시하고, 이를 map 상에 시각적으로 보여준다. quit 메뉴를 선택하면 프로그램이 종료된다.

Interface 서브시스템은 아래와 같은 클래스들로 구성되어 있다.

- 인터페이스 IDesign

  창의 디자인과 관련된 인터페이스이다. "WindowDesign" 클래스가 구현한다.

- 클래스 WindowDesign

  이 클래스는 새 창을 띄우는 클래스인 ShowMenu, ShowMapData, SaveData, ShowRobotMovement 클래스의 부모 클래스이다. 창의 크기 설정, 레이블 생성, 텍스트 박스 생성, 버튼 생성, 위젯들의 속성 변경, 레이아웃에 위젯 추가 등 창의 디자인과 관련된 기능을 수행한다. 이 클래스는 QWidget 클래스를 상속받고, IDesign 인터페이스를 구현한다..

  __init__(self) : 부모 클래스인 QWidget 클래스의 생성자를 호출한다.

  setWindowDesign(self, title, image, width, height, xPos=0, yPos=0) : 제목이 title, 아이콘이 image 이고 가로 길이와 세로 길이가 각각 width, height 인 창을 화면 왼쪽 위 꼭짓점에서 xPos 만큼 아래로, yPos 만큼 오른쪽으로 떨어진 곳에 생성한다.

  makeLabel(self, labelNameList) : labelNameList 에는 각 레이블에 들어갈 텍스트가 들어있다. 각 텍스트를 이용하여 레이블을 생성하고 이것들을 새로운 리스트 labelList 에 담아 리턴한다.

makeTextLine(self, num, isEditable, baseText=['']) : num 개의 TextLine 을 만든다. 이 텍스트 라인은 맵 정보를 입력 받을 때나 저장된 맵 데이터를 표시할 때 사용한다. isEditable 이 True 이면 TextLine 은 수정 가능하고, False 이면 수정 불가능하다. baseText 에는 num 개의 문자열이 들어있으며, 이는 TextLine 을 만들 때 매개변수로 사용한다. baseText 에 값이 들어오지 않을 경우 baseText 에 num 개의 빈 문자열을 넣어서 TextLine 을 만들 때 사용한다.

_makeGroupBox(self, title) : 이 메소드는 WindowDesign 클래스 내부에서만 사용한다. WindowDesign 의 setLayoutOnGroupBox 메소드에서 사용하며, title 을 제목으로 하는 QGroupBox 를 생성하여 리턴한다. 제목은 가운데 정렬하여 표시한다.

setLayoutOnGroupBox(self, title, customLayout) : _makeGroupBox 에 매개변수 title 을 전달하여 QGroupBox 를 생성하고, 여기에 customLayout 을 적용한 뒤 생성된 QGroupBox 를 리턴한다.

makeButton(self, name, clickFunction) : CustomButton 클래스를 통해 새로운 버튼을 생성한다. 버튼은 텍스트로 name 을 가지고 있으며, 버튼 클릭 이벤트 발생시에 clickFunction 메소드와 연결된다.

setCustomStyleSheet(self, widget, styleSheet) :

widget 에 PyQt5 의 setStyleSheet() 메소드에 styleSheet 를 매개변수로 전달하여 디자인을 적용한다. styleSheet 는 CSS 문법을 가지는 문자열 형태로 되어있다. widget 은 단일 대상 또는 widget 들의 리스트 형태로 들어올 수 있으며, type(widget)을 통해 단일 대상인지 리스트인지를 결정하고 리스트이면 리스트의 원소들에 각각 setStyleSheet()를 적용시켜준다.

addWidgetOnLayout(self, layout, widget, *x) : layout 에 widget 을 추가시켜준다. layout 이 QGridLayout 일 경우 x 에는 두개의 숫자가 들어가고, 그 외의 다른 레이아웃일 경우 x 에 아무것도 들어가지 않는다. x 의 길이가 0 이면 layout.addWidget(widget)을, 2 이면 layout.addWidget(widget, x[0], x[1])을 수행한다.

addLayoutOnLayout(self, baseLayout, addLayout, ratio=1) : baseLayout 에 addLayout 을 추가시킨다. ratio 는 baseLayout 에 여러 개의 레이아웃을 추가시킬 때 그들 간의 화면 차지 비율을 결정한다..

design(self) : 추상메소드이다. WIndowDesign 을 상속받는 클래스들은 이 메소드를 구현해서 사용하는 것으로 자신만의 창을 꾸밀 수 있다.

showMessage(self, head, message, toggle) : PyQt5 의 QmessageBox 클래스를 이용하여 제목이 head 인 message 를 띄워준다. toggle 값이 0 일 경우 메시지 박스의 확인 버튼을 누른 뒤 창을 닫지 않고, 1 일 경우 닫는다.

- 클래스 CustomButton

QPushButton 을 상속받는 클래스로, 버튼에 추가적인 애니메이션을 적용시키기 위한 클래스이다. 애니메이션은 두 개의 색으로 구성된 그라데이션으로 이루어진다. ADD-ON 프로그램에 사용되는 버튼들은 전부 이 클래스의 인스턴스이다.

__init__(self) : 부모 클래스인 QPushButton 클래스의 생성자를 호출한다. 버튼의 최소 사이즈, 애니메이션에서 사용할 두 색 startColor, endColor 의 RGB 를 설정하고 QtCore.QVariantAnimation 클래스의 인스턴스를 생성하여 private 클래스 변수 _animation 에 저장한다. 인자로 valueChanged, startValue, endValue, duration 이 주어진다. valueChanged 에는 애니메이션에 사용할 메소드가 들어가며 startValue 와 endValue 는 각각 애니메이션의 시작 지점과 끝 지점을 결정한다. duration 은 애니메이션 재생 속도를 결정한다.

_customAnimate(self, value) : 변수 _animation 의 매개변수로 주어진다. value 는 마우스가 버튼 위에 있을 때 startValue 에서 endValue 까지 점차적으로 증가하며, 마우스가 버튼에서 떨어지면 endValue 에서 startValue 까지 점차적으로 감소한다.

enterEvent(self, event) : 마우스가 버튼을 가리킬 때 재생되는 애니메이션이다. CustomButton 이 상속받는 QButton 의 부모클래스인 QWidget 에 구현되어 있는 메소드를 오버라이딩한 것이다.

leaveEvent(self, event) : 마우스가 버튼에서 떨어질 때 재생되는 애니메이션이다. CustomButton 이 상속받는 QButton 의 부모클래스인 QWidget 에 구현되어 있는 메소드를 오버라이딩한 것이다.

- 클래스 ShowMenu

이 클래스는 WindowDesign 클래스를 상속받는다. 프로그램을 처음 실행했을 때 나오는 메뉴와 관련된 역할을 수행한다.

- __init__(self) : 부모 클래스인 WindowDesign 클래스의 생성자를 호출하고 design() 메소드를 호출한다.

- design(self) : 초기 메뉴 창을 디자인해주는 메소드이다. 부모 클래스인 WindowDesign 의 추상 메소드를 구현해서 사용한다.

- saveData(self) : save data 메뉴를 클릭했을 때 실행된다. SaveData 클래스의 객체를 생성한다.

- showMapData(self) : show map data 메뉴를 클릭했을 때 실행된다. MapData 클래스에 맵 데이터가 저장되어 있는지 확인하기 위해 MapData 클래스의 getMapSize() 메소드를 호출해보고, 에러가 발생할 경우 WindowDesign 클래스의 showMessage 메소드를 이용하여 오류 메시지를 띄운다. 에러가 발생하지 않을 경우 ShowMapData 클래스의 인스턴스를 생성한다.

- showResult(self) : show result 메뉴를 클릭했을 때 실행된다. MapData 클래스에 맵 데이터가 저장되어 있는지 확인하기 위해 MapData 클래스의 getMapSize() 메소드를 호출해보고, 에러가 발생할 경우 WindowDesign 클래스의 showMessage 메소드를 이용하여 오류 메시지를 띄운다. 에러가 발생하지 않을 경우 ShowResult 클래스의 인스턴스를 생성한다.

- 클래스 ShowMapData

이 클래스는 WindowDesign 클래스를 상속받는다.

- __init__(self) : WindowDesign 클래스의 생성자를 호출하고 MapData 로부터 맵 크기, stat spot, object spot, hazard spot 정보를 받는다.

- design(self) : 생성자에서 받은 맵 크기, start spot, object spot, hazard spot 정보를 새로운 창에 띄워서 사용자에게 보여준다.

- 클래스 SaveData

이 클래스는 WindowDesign 클래스를 상속받는다.

- __init__(self) : WindowDesign 클래스의 생성자를 호출하고 design() 메소드를 호출한다.

-design(self) : WindowDesign 클래스의 메소드들을 이용하여 새로운 창을 띄운다. 이 창에는 map 크기, 시작 지점, 목표 지점, 위험 지점을 입력할 수 있는 4 개의 textLine 과 save, cancel 버튼이 존재한다. 각 데이터들을 입력받고 save 버튼을 누르면 saveInputData() 메소드를 실행한다. cancel 버튼을 누르면 메인 메뉴로 돌아간다. 각 textLine 을 저장하는 클래스 변수 textLineList 를 만든다.

- saveInputData(self) : 입력값들을 적절하게 처리하여 사용할 수 있는 좌표값으로 만든다. 이 과정이 끝나면 각 입력값들이 유효한지 inputValueCheck() 메소드를 통해 확인한다. inputValueCheck() 메소드의 반환값이 True 이면 오류 메시지를 띄운다. 반환값이 False 일 경우 입력값들을 MapData 클래스의 초기화에 사용한다.

- inputValueCheck(self, mapSize, start, spot, hazard) : 매개변수로 들어온 4 개의 값들을 이용하여 아래의 테스트를 진행한다.

1. mapSize 에 음수값이 들어올 경우 error
2. hazard spot 이나 object spot 이 맵 밖으로 나갈 경우 error

3. hazard spot, object spot 둘 다에 속하는 점이 있을 경우 error

4. 시작 지점이 맵 밖으로 나가거나 spot 또는 hazard 에 속할 경우 error

- 클래스 ShowRobotMovement

초기 메뉴 화면에서 'Show robot movement' 항목을 선택했을 때 이 클래스의 인스턴스가 생성된다. 이 클래스는 WindowDesign 클래스를 상속받는다.

- __init__(self) : WindowDesign 클래스의 생성자를 호출하고 init() 메소드를 실행한다. 그 후 setInitialInfo() 메소드를 실행하여 여러 변수들을 초기화 시킨다(자세한 내용은 밑에서 setInitialInfo() 메소드를 설명할 때 다시 설명함). 이 작업이 끝나면 주어진 hidden hazard spot 갯수와 hidden color blob spot 갯수를 매개변수로 주고 generateRandomSpot 메소드를 실행하여 무작위 지점에 hidden hazard spot 과 hidden color blob spot 을 만든다. 생성 작업이 끝나면 DrawMap 클래스의 인스턴스를 생성하고 design() 메소드를 실행한다.

- setInitialDirection(self, secondSpot, firstSpot) : secondSpot 과 firstSpot 의 차이를 통해서 로봇의 처음 방향을 구한 뒤 리턴한다.

- setInitialInfo(self) : MapData 클래스에서 맵 크기, 시작 위치, hazard spot, object spot 정보를 받아와서 클래스 변수에 할당해준다. objCnt 에는 object spot 의 개수를 할당해주고 PathController 와 RobotController 의 인스턴스도 생성해준다. PathController 인스턴스를 통해 경로를 생성한 뒤 그 경로를 클래스 변수 path 에 저장해준다.

- design(self) : Show Robot Movement 창 디자인을 담당한다. 창 왼쪽에는 DrawMap 클래스의 인스턴스에서 받아온 맵이 존재하며, 오른쪽에는 현재 로봇의 위치, 방향, 남은 object spot 의 개수와 각 아이콘의 의미를 나타내는 표가 존재한다. 창의 아래쪽에는 start 버튼과 return to menu 버튼이 존재한다. start 버튼은 controlRobotMovement 메소드와 연결되어 있다. return to menu 버튼을 누르면 창이 닫힌다.

- changePath(self) : Robot 서브 시스템의 클래스 PathController 의 객체 ctrlPath 를 이용하여 현재 위치에서 시작하는 새로운 경로를 만든다. ctrlPath 의 createPath 메소드에 현재 로봇 위치와 RobotController 클래스의 인스턴스인 ctrlRobot 를 통해 얻을 수 있는, 로봇이 지금까지 발견한 hidden hazard spot 리스트를 매개변수로 준다. 만들어진 경로 정보를 ctrlPath 의 getPath 메소드를 통해 받아서 클래스 변수 path 에 저장한다. DrawMap 인스턴스의 메소드 drawPath 를 실행하여 기존 경로를 맵에서 지우고 새 경로를 그린다.

- showMovingProcess(self) : 창에서 start 버튼을 눌렀을 때 실행되는 메소드이다. 로봇의 이동과정을 보여주는 역할을 한다. 동작 과정은 아래와 같다.

  1. 인근 지역에 숨겨진 color spot 과 로봇이 보고있는 방향 바로 앞에 숨겨진 hazard spot 을 맵에 표시한다.
  2. comandMovementAndCangePosInfo 메소드를 통해 움직임이 끝난 로봇의 위치와 방향을 받아서 기존에 있던 로봇을 지우고 움직인 위치에 새로 그린다.
  3. 로봇이 object spot 에 도달했다면 objCnt 를 1 줄인다.
  4. objCnt 가 0 이 되면 완료 메시지를 띄우고 메소드가 종료된다.

- commandMovementAndChangePosInfo(self, hazardFound) : 로봇의 움직임을 지시하고 움직임이 끝난 뒤 현재 로봇의 위치 정보, 방향 정보를 받아 curPosition, curDirection 을 수정한다. DrawMap 클래스의 인스턴스 drawMap 을 이용하여 기존에 맵에 있던 로봇을 지우고 움직인 위치에 로봇을 다시 그린다. 만일 로봇이 두 칸 앞으로 이동했다면 changePath() 메소드를 호출해 기존 경로를 지우고 새로운 경로를 그린다.

- generateRandomSpot(self, hazardNum=-1, cbNum=-1) : 랜덤한 hazard spot 과 color blob spot 을 생성한다. 먼저 MapData 클래스의 get HazardSpot() 메소드와 getObjectSpot(), getStartSpot() 메소드를 통해 좌표들을 받고, 이 좌표들을 제외한 무작위 좌표에 hazard spot 과 color blob spot 을 각각 hazardNum, cbNum 개 만큼 생성한다. hazardNum 이나 cnNum 이 음수일 경우 0~(맵의 가로 길이 + 맵의 세로 길이) / 2 개 사이로 생성한다. 이때, 랜덤 생성한 hazard spot 이 object spot 에 도달하지 못하도록 막고있는 경우 막고있는 hidden hazard spot 들 중 무작위로 하나를 골라 삭제한다. 최종 hazard spot 과 color blob spot 을 MapData 클래스의 setHiddenData(hazard, cb) 메소드를 통해 저장한다.

- 인터페이스 IDrawMap

  맵을 초기화하고 맵에 로봇 아이콘과 경로를 그리며, 기존에 존재하던 아이콘을 지우고 새로 그리는 역할을 한다. 또한 다 만든 맵을 canvas 에 담아 리턴하는 역할을 한다. DrawMap 클래스가 구현한다.

- 클래스 DrawMap

  ShowRobotMovement 가 띄우는 창에 사용할 맵을 생성하고 그 위에서 그림을 그리는 클래스이다.

  - __init__(self, mapSize, curPosition, curDirection, hazardSpot, objectSpot, hiddenHazardSpot, hiddenColorBlobSpot, path) : 매개변수로 주어진 정보들을 클래스 변수에 저장하고, robotImage, arrowImage, drawedLines, enlarge 등의 변수를 선언한다. robotImage 와 arrowImage 는 맵 상에 그려져 있는 로봇과 화살표 아이콘을 객체로 나타낸 것이며, drawedLines 는 현재 그려져 있는 경로들의 객체의 리스트이다. enlarge 는 맵을 확대할지 말지를 결정하며, 맵의 가로, 세로 길이가 모두 10 보다 클 때 True 가 된다. 변수들을 모두 선언하면 drawInitialMap 메소드를 실행한다. 매개변수로는 curPosition, curDirection, path 를 준다.

  - drawInitialMap(self, curPosition, curDirection, path) : 맵을 표현할 때 사용할 matplotlib 의 figure 클래스의 인스턴스 fig 를 생성한다. 이대로는 PyQt5 와 연동할 수 없으므로 fig 를 매개변수로 FigureCanvasQTAgg (이 프로그램에서는 FigureCanvas) 객체를 생성한다. drawBorder() 메소드를 실행하여 plot 상에 맵 외곽선을 긋고 차례대로 drawRobot(), drawInitialMapIcon(), drawPath(path) 메소드를 실행하여 로봇, hazard spot, object spot, hidden hazard spot, hidden color blob spot, 초기 경로를 맵 상에 그린다.

- drawInitialMapIcon(self) : MapData 에 저장되어 있는 정보를 이용하여 hazard spot 과 object spot 을 맵 상에 표시한다. 클래스 리스트 변수 hiddenHazardSpotInstance, hiddenColorBlobSpotInstance, objectSpotInstance 를 생성하는데, 각각의 리스트들은 hidden hazard spot, hidden color blob spot, object spot 의 좌표, 객체 순서쌍을 원소로 가지고있다. generateRandomSpot() 메소드를 통해 생성한 hidden hazard spot 과 hidden color blob spot 도 맵 상에 흐리게 표시한다.

- drawBorder(self) : 클래스 변수로 저장한 맵의 크기 mapSize 를 이용하여 맵의 외곽선을 그린다.

- imageScatter(self, x, y, image, zoom=1, ax=None) : x, y 좌표에 경로 image 에 있는 이미지를 그린다. image 는 이미지 파일의 경로, zoom 은 이미지 확대 배율, ax 는 이미지를 그릴 matplotlib 의 plot 을 나타낸다. 메소드의 작업 과정은 다음과 같다.

    1. 먼저 matplotlib.pyplot 의 imread 메소드를 이용하여 이미지를 2 차원의 numpy 배열로 변환하여 이를 로컬 변수 image 에 저장한다.

    2. matplotlib 의 클래스 OffsetImage 의 인스턴스 im 을 생성한다. 이때 변수 image 와 zoom 을 매개변수로 준다.

    3. matplotlib 의 클래스 AnnotationBBox 의 인스턴스 artist 를 생성한다. 이때 위에서 생성한 im 과 좌표 (x, y), frameon=False 를 매개변수로 준다. AnnotationBBox 클래스는 plot 원래 위에 메모를 적기위한 용도로 사용된다. 여기에 이미지를 넣어 마치 해당 좌표의 점에 이미지를 넣은 것처럼 사용하는 것이다. frameon 에 True 를 주면 이미지에 테두리가 생긴다.

    4. plot ax 에 위에서 생성한 artist 를 추가하고 그 결과를 리턴한다.

- drawImage(self, x, y, imagePath) : 맵 상의 좌표 (x, y)에 imagePath 에 있는 이미지를 그린다.

- eraseAndDrawImage(self, spot, imageInstanceList, imagePath) : 기존에 spot 위치에 있던 이미지를 지우고 imagePath 에 있는 이미지로 대체한다.

- drawRobot(self, position, direction) : 로봇의 위치를 바탕으로 로봇을 맵에 그린다. enlarge 값이 True 일 경우 맵을 확대해서 표시한다.

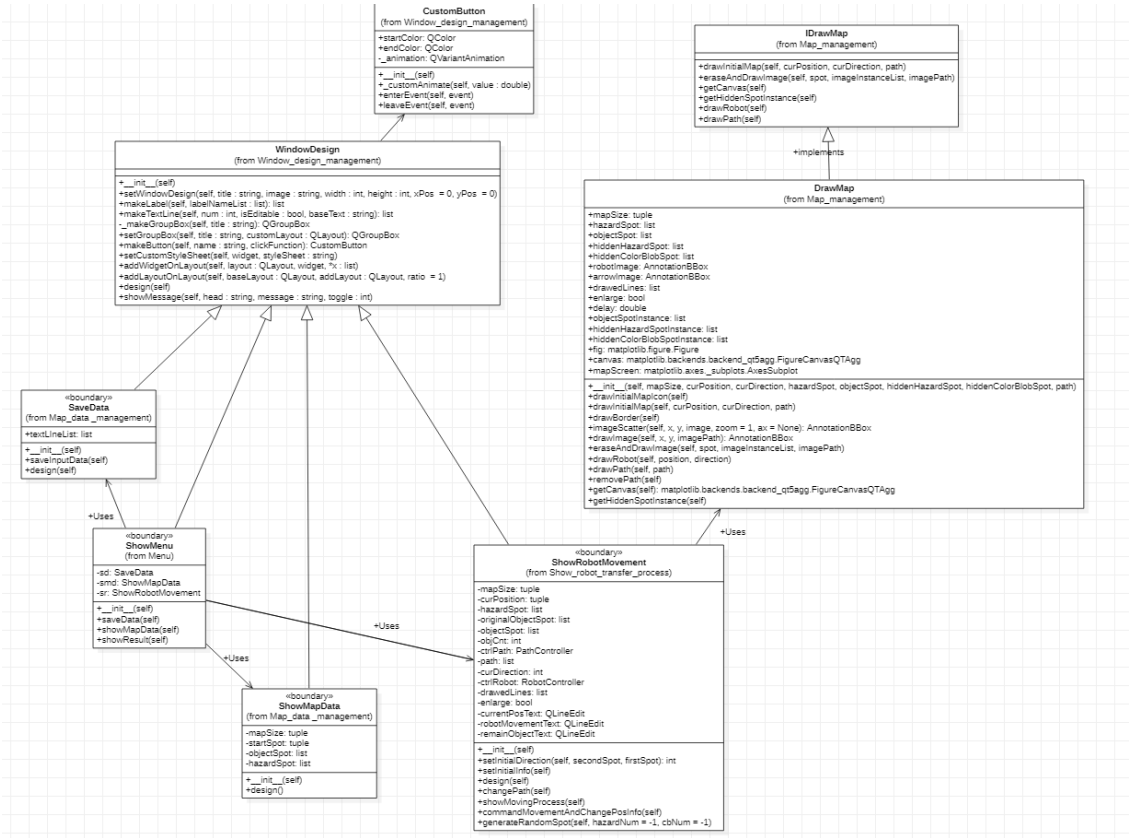
- drawPath(self, path) : 로봇이 지나갈 경로를 맵 상에 그린다. 경로 값은 점들로 이루어져 있는데, 각 점들 사이마다 직선을 모두 그으면 맵 크기가 커졌을 때 시간이 너무 오래 걸려 하나의 직선 위에 있는 점들은 양 끝점을 잇는 식으로 해결했다.

- _removePath(self) : 화면에 보이는 경로를 지운다. 이 때 맵 외곽선도 같이 지워지므로 맵 외곽선을 새로 그려준다. 이 메소드는 DrawMap 클래스 안에서만 사용된다.

- getCanvas(self) : PyQt5 의 창에 맵을 추가하기 위해 FigureCanvasQTAgg 형 변수 canvas 를 리턴한다.

- getHiddenSpotInstance(self) : drawIntialMapIcon 메소드에서 생성한 변수 hiddenHazardSpotInstance, hiddenColorBlobSpotInstance, objectSpotInstance 를 리턴한다.

3.3.1 메인 클래스도



ShowMenu

- 프로그램 초기 실행 시 메뉴창을 띄워준다.
- 각각의 메뉴에 따라 map data 저장, 저장된 map data 출력, mobile robot 이동 과정 출력과 같은 기능을 수행한다. show robot movement 메뉴 선택 시 ShowRobotMovement 클래스의 인스턴스를 생성한다.

SaveData

- Operator 에게 맵 정보를 입력 받아서 이를 시스템에 저장한다. 저장한 정보는 Database 서브시스템의 MapData 클래스를 통해 접근할 수 있다.

ShowMapData

- Database 서브시스템에 저장된 맵 정보를 Operator 에게 보여준다.

ShowRobotMovement

- 메뉴에서 Show robot movement 를 선택했을 때 객체가 생성된다. 격자 그래프를 이용하여 맵을 띄워주며, 맵 아랫부분에는 next 버튼과 return to menu 버튼이 존재한다.

- return to menu 버튼을 누를 시 메인 메뉴로 돌아간다.

- start 버튼을 누르면 mobile robot 의 움직임을 일정 시간 간격으로(기본값 0.5 초) 보여준다. 로봇은 경로를 따라 움직이면서 hidden hazard spot 이나 hidden color blob spot 을 감지할 수 있으며, hidden hazard spot 을 발견할 때마다 WindowDesign 클래스의 showMessage 메소드를 통해 메시지 창을 띄워 operator 에게 알려주고 map 에 표시한다. 로봇이 모든 목표 지점에 도달하면 모든 목표지점에 도달했다는 메시지를 띄운 뒤 메인 메뉴로 돌아간다.

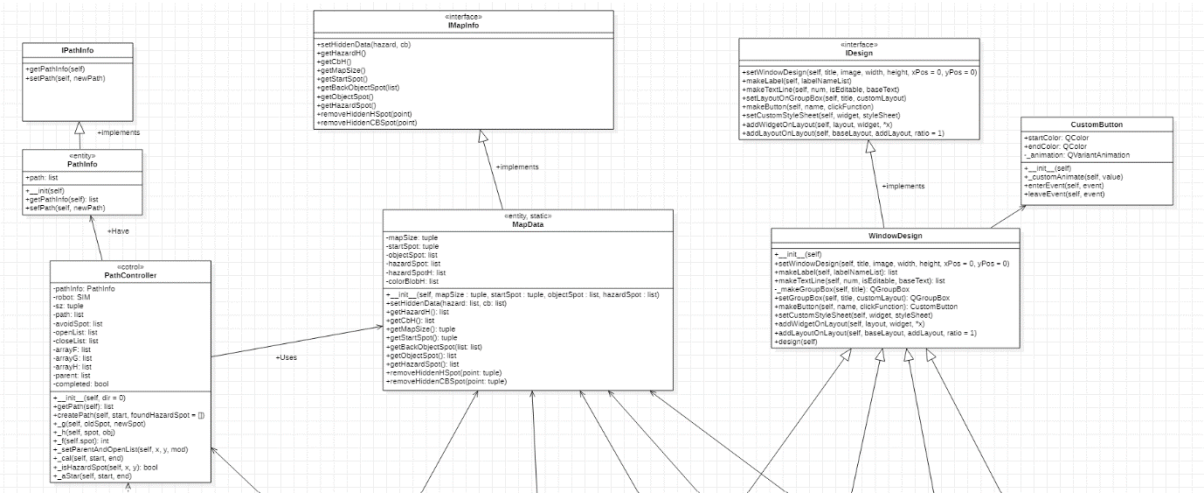
WindowDesign

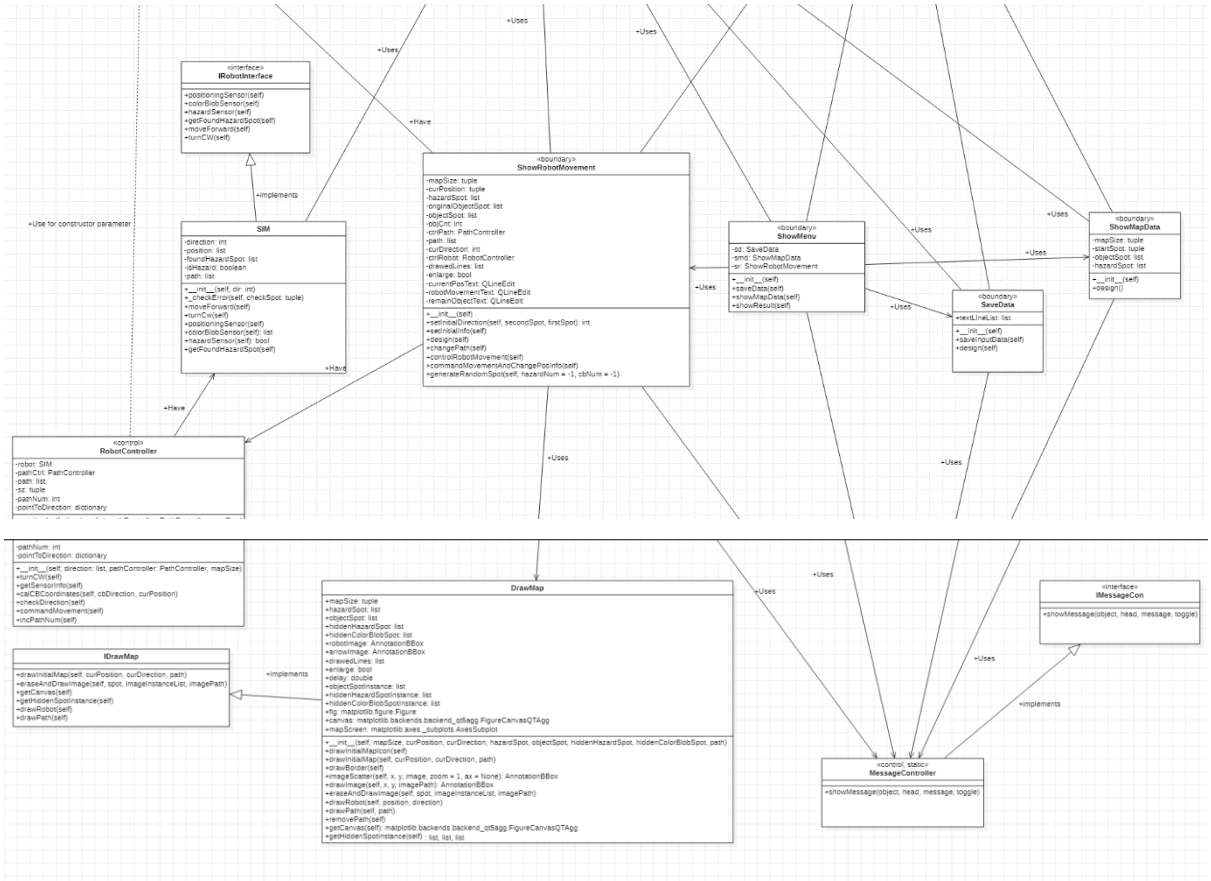
- 새 창을 띄우는 클래스인 ShowMenu, ShowMapData, SaveData, ShowRobotMovement 클래스의 부모 클래스이다. 창의 크기 설정, 레이블 생성, 텍스트 박스 생성, 버튼 생성, 위젯들의 속성 변경, 레이아웃에 위젯 추가 등 창의 디자인과 관련된 기능을 수행한다.

CustomButton

- 새 창을 띄우기 위해 사용된 버튼에 추가적인 애니메이션을 적용시킨다.

3.4 전체 시스템 클래스도





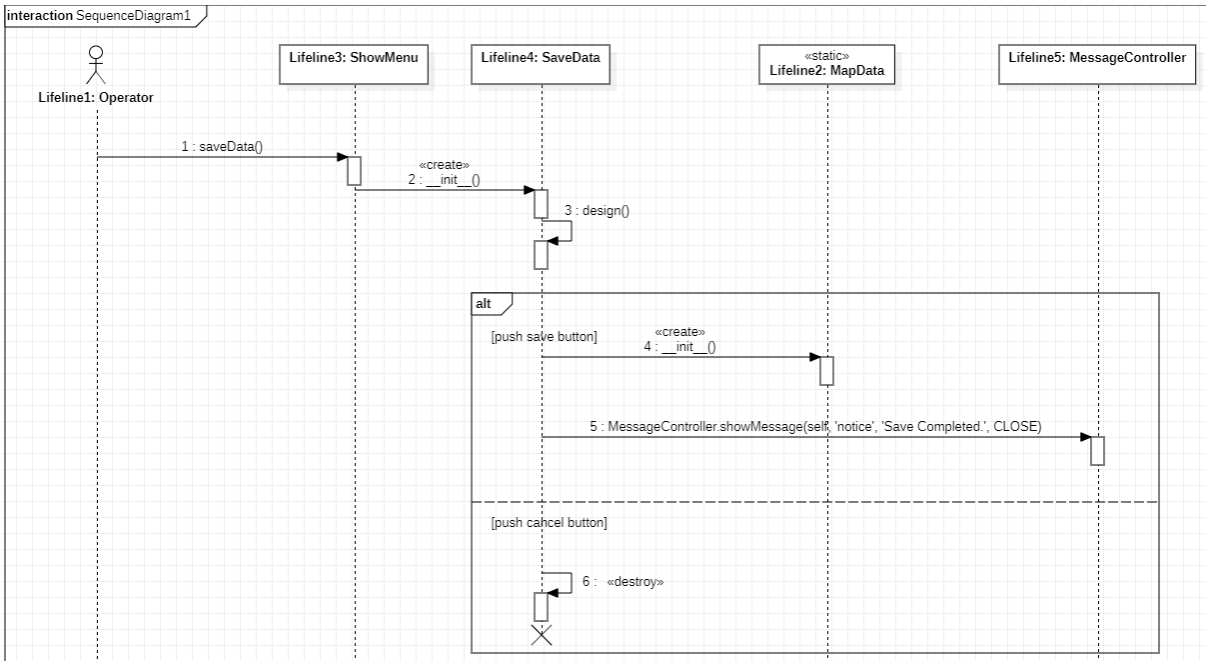
MapData 클래스는 정적 클래스이며, SIM, PathController, ShowRobotMovement, ShowMenu, SaveData, ShowMapData 클래스에서 공유한다.

RobotController 클래스의 인스턴스가 생성될 때 pathController 클래스의 인스턴스를 생성자의 매개변수로 사용한다. SIM 클래스의 인스턴스를 생성한다.

ShowMenu, ShowMapData, SaveData, ShowRobotMovement 클래스는 WindowDesign 클래스를 상속받는다.

4. 교류도

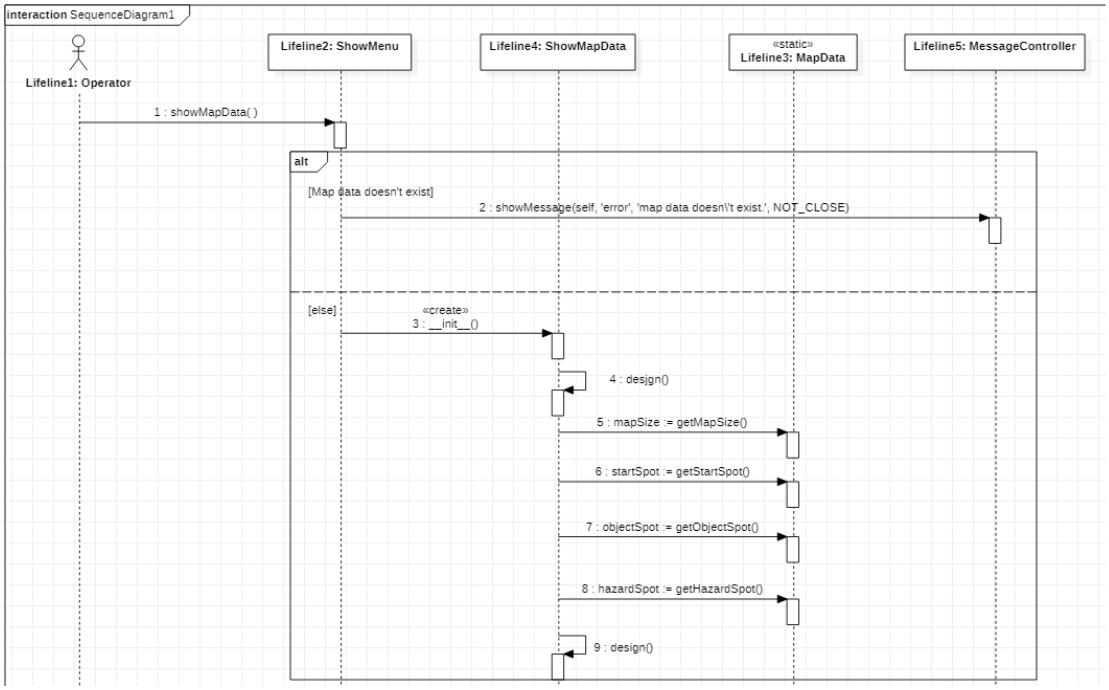
4.1 Use Case : Save initial map information



Operator 가 메뉴에서 save data 항목을 선택하면 saveData() 메소드가 실행된다. ShowMenu 클래스는 맵 정보들을 입력 받는 창을 띄우고 유저가 Save 버튼을 누르면 입력한 데이터들일 유효한지 확인한다. 유효하면 이들 정보를 매개변수로 MapData 객체를 생성하여 정보를 저장한다. 유효하지 않으면 데이터를 다시 입력해달라는 메시지를 띄운다. 저장이 완료되면 showMessage() 메소드를 통해 완료 메시지를 띄운다. 유저가 cancel 버튼을 누르면 메인 메뉴로 돌아간다는 메시지를 띄우고 메인 메뉴로 돌아간다.

1. showMapData() 메소드는 메뉴에서 show map data 항목을 눌렀을 때 실행된다. 유저에게 여러가지 맵 관련 정보들을 입력 받고 이를 MapData 객체에 저장한다.
2. ShowMessage() 메소드는 상황에 맞는 메시지를 메시지 박스에 담아서 화면에 출력한다.

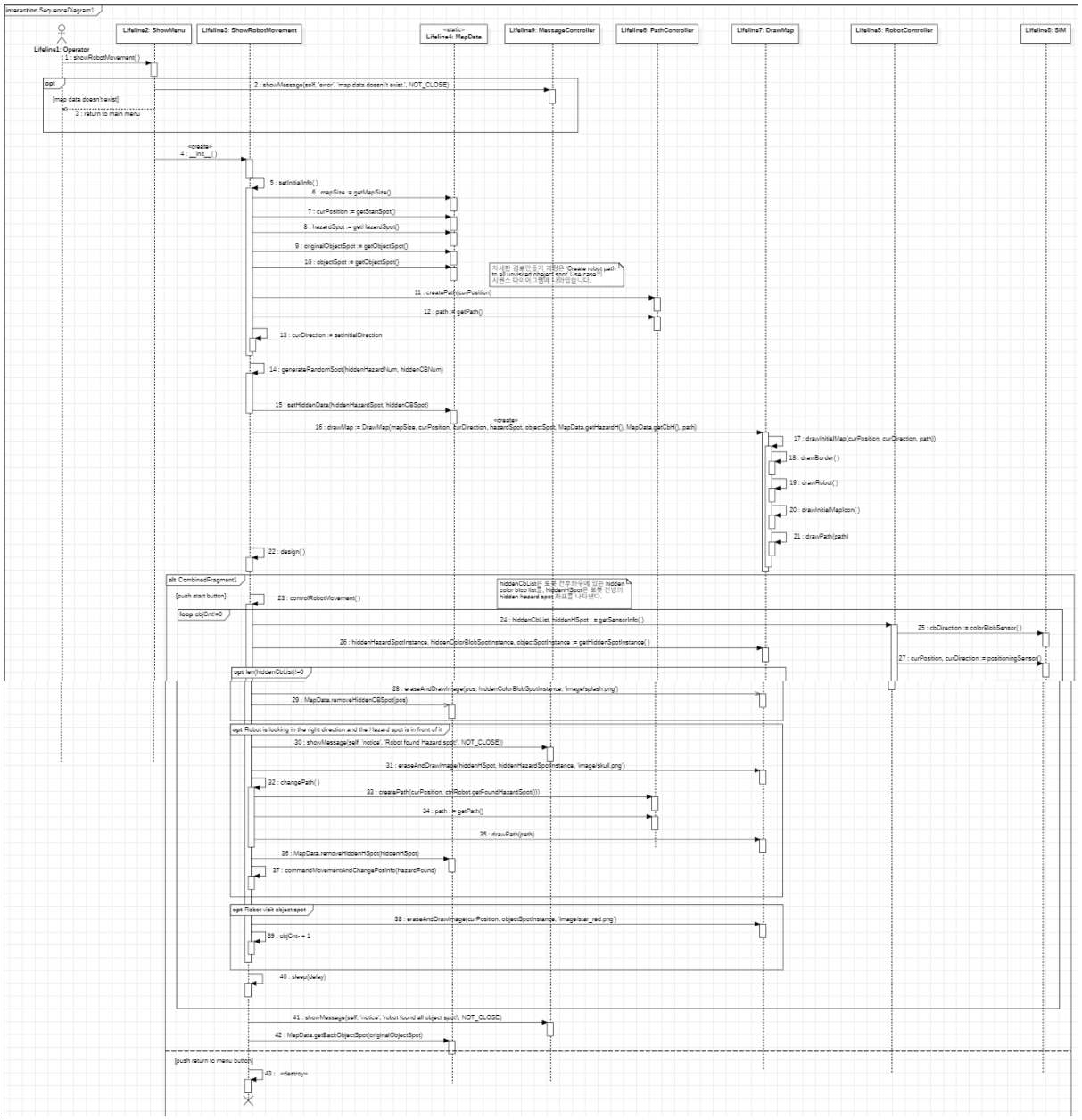
4.2 Use Case : Show saved map data

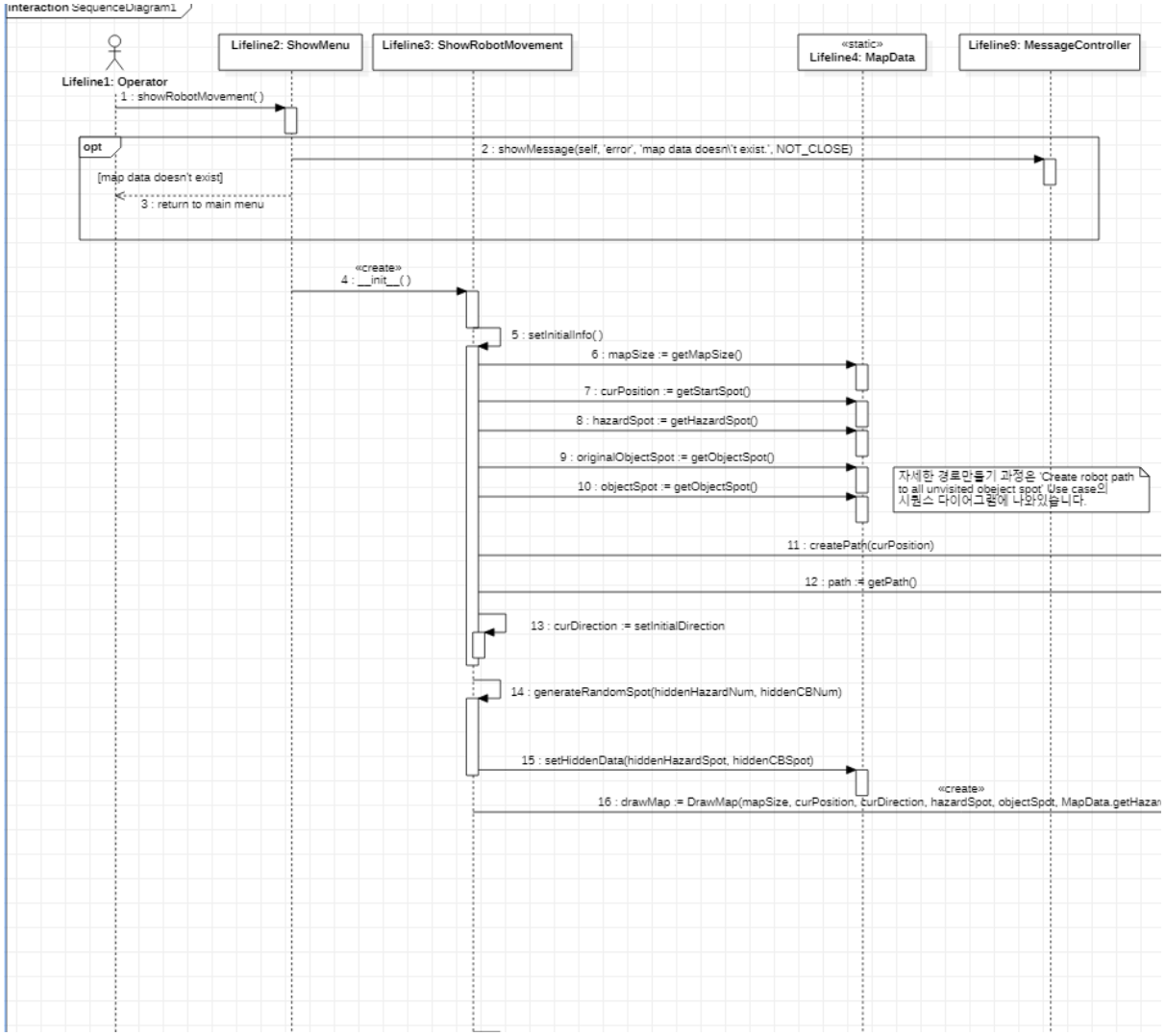


Operator 가 메뉴에서 show map data 를 선택하면 showMapData()가 실행된다. 먼저 저장된 맵 정보가 있는지 확인하고 존재하지 않을 경우 showMessage()를 통해 에러 메시지를 출력하고 돌아간다. 존재할 경우 get 메소드 형태로 맵 정보를 받아와서 출력한다.

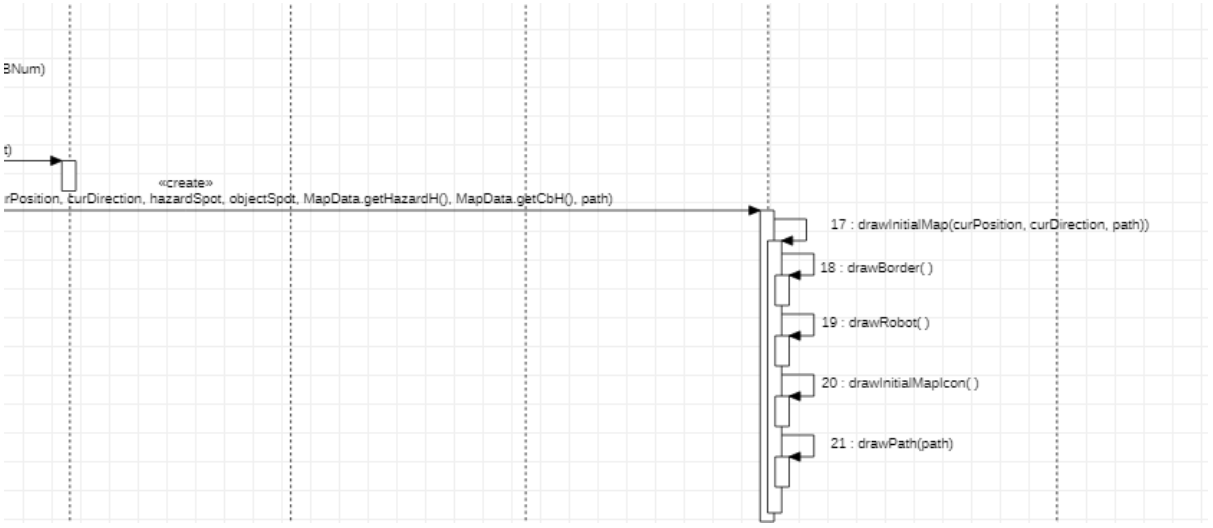
1. showMapData() 메소드는 메뉴에서 show map data 항목을 눌렀을 때 실행된다. 유저에게 입력 받은 맵 정보를 받아 출력한다.
2. ShowMessage() 메소드는 상황에 맞는 메시지를 메시지 박스에 담아서 화면에 출력한다
3. getMapSize(), getStartSpot(), getObjectSpot(), getHazardSpot() 메소드는 MapData 클래스에 저장되어 있는 맵 정보를 받아온다.

4.3 Use Case : Show robot movement to all object spot

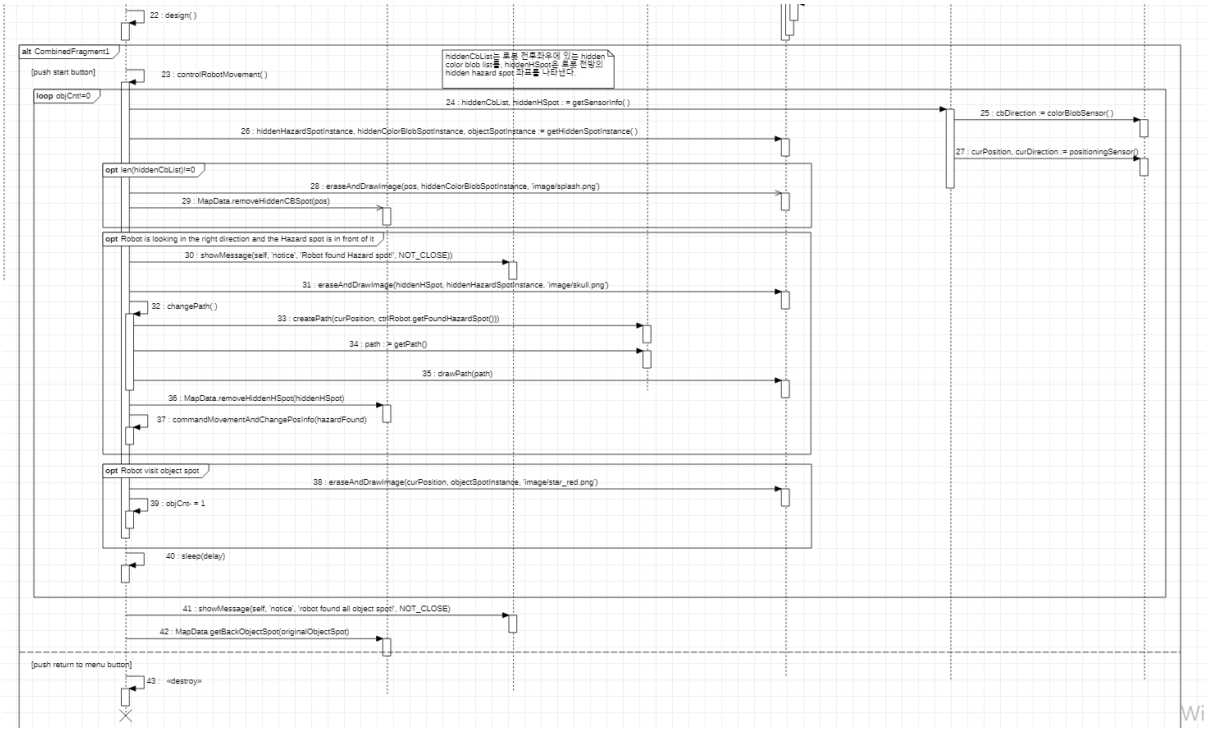

interaction SequenceDiagram1
Lifeline1: Operator
Lifeline2: ShowManu
Lifeline3: ShowRobotMovement
«static»
Lifeline4: MapData
Lifeline9: MassageController
Lifeline6: PathController
Lifeline7: DrawMap
Lifeline5: RobotController
Lifeline8: SIM
1 : showRobotMovement( )
opt
2 : showMassage(self, 'error', 'map data doesn't exist.', NOT_CLOSE)
[map data doesn't exist]
3 : return to main manu
«create»
4 : __init__( )
5 : setInitialInfo( )
6 : mapSize = getMapSize()
7 : curPosition = getStartSpot()
8 : hazardSpot = getHazardSpot()
9 : originalObjectSpot = getObjectSpot()
10 : objectSpot = getObjectSpot()
자세한 경로만들기 내용은 'Create robot path to all unvisited object spot' Use case에 서술되 있습니다.
11 : createPath(curPosition)
12 : path = getPath()
13 : curDirection = setInitialDirection
14 : generateRandomSpot(hiddenHazardNum, hiddenCBNum)
15 : setHiddenData(hiddenHazardSpot, hiddenCBSpot)
«create»
16 : drawMap = DrawMap(mapSize, curPosition, curDirection, hazardSpot, objectSpot, MapData.getHazardH(), MapData.getCbH(), path)
17 : drawInitialMap(curPosition, curDirection, path))
18 : drawBorder( )
19 : drawRobot( )
20 : drawInitialMapIcon( )
21 : drawPath(path)
22 : design( )
alt CombinedFragment1
hiddenCbList는 로봇 전후좌우에 있는 hidden color blob list를, hiddenHSpot은 로봇 전방의 hidden hazard spot 가르킬 내용입니다.
[push start button]
23 : controlRobotMovement( )
loop objCnt!=0
24 : hiddenCbList, hiddenHSpot : = getSensorInfo( )
25 : cbDirection := colorBlobSensor( )
26 : hiddenHazardSpotInstance, hiddenColorBlobSpotInstance, objectSpotInstance := getHiddenSpotInstance( )
27 : curPosition, curDirection := positioningSensor( )
opt len(hiddenCbList)!=0
28 : eraseAndDrawImage(pos, hiddenColorBlobSpotInstance, 'image/splash.png')
29 : MapData.removeHiddenCBSpot(pos)
opt Robot is looking in the right direction and the Hazard spot is in front of it
30 : showMassage(self, 'notice', 'Robot found Hazard spot', NOT_CLOSE))
31 : eraseAndDrawImage(hiddenHSpot, hiddenHazardSpotInstance, 'image/skull.png')
32 : changePath( )
33 : createPath(curPosition, ctrlRobot.getFoundHazardSpot()))
34 : path := getPath()
35 : drawPath(path)
36 : MapData.removeHiddenHSpot(hiddenHSpot)
37 : commandMovementAndChangePosInfo(hazardFound)
opt 'Robot visit object spot'
38 : eraseAndDrawImage(curPosition, objectSpotInstance, 'image/star_red.png')
39 : objCnt- = 1
40 : sleep(delay)
41 : showMassage(self, 'notice', 'robot found all object spot', NOT_CLOSE)
42 : MapData.getBackObjectSpot(originalObjectSpot)
[push return to manu button]
43 : «destroy»



첫 번째 이미지.

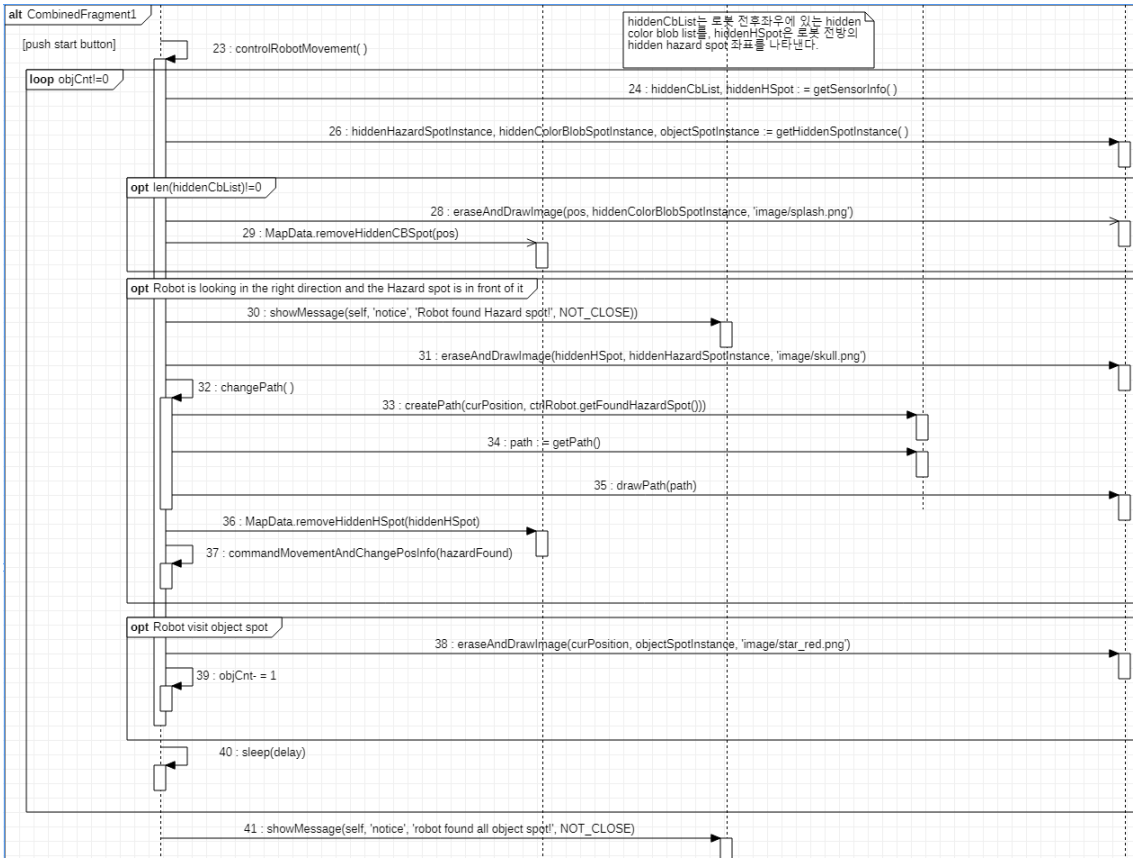


두 번째 이미지. 첫 번째 이미지가 나타내는 부분에서 더 오른쪽을 찍은 것이다.



세 번째 이미지. 첫 번째 이미지와 두 번째 이미지보다 더 아랫부분을 찍은 것이다.

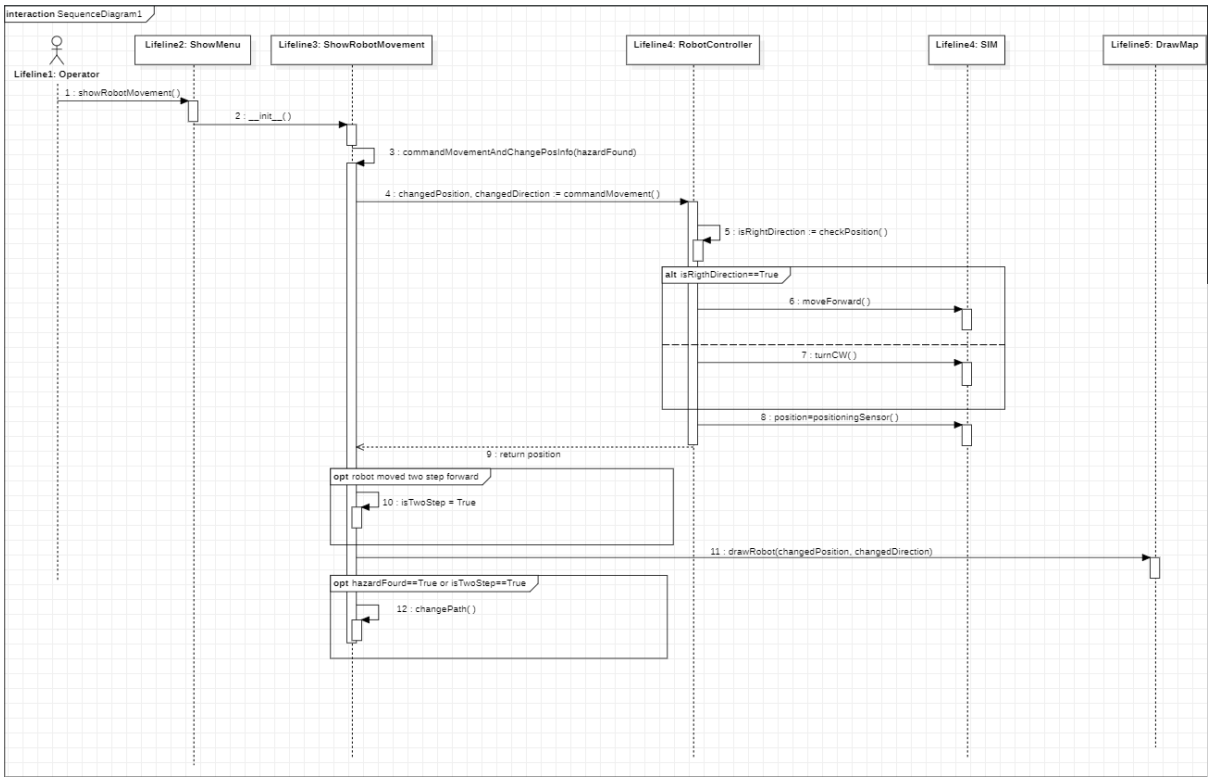
로봇의 이동과정을 그리는 부분을 확대한 이미지



Operator 가 show robot movement 를 선택했을 때 showRobotMovement() 메소드가 실행된다. 저장되어있는 맵 정보가 있는지 확인하고 없으면 에러 메시지를 띄워주고 메인으로 돌아간다. 존재할 경우 setInitialInfo()를 통해 맵 정보를 MapData 로부터 받아오고 createPath()를 통해 만들어진 초기경로를 getPath() 메소드로 받아와서 저장한다. 그 후 generateRandomSpot(self, hazardNum=-1, cbNum=-1) 메소드를 통해 무작위로 숨겨진 spot 들을 생성하고 MapData 객체의 setHiddenData(hazard, cb) 메소드를 통해 해당 객체에 저장한다. Start 버튼을 누르면 controlRobotMovement() 메소드를 통해 로봇을 움직인다. 중요지점인 objCnt 가 0 이 될때까지 반복한다. 위험지점을 발견할 경우 changePath(), drawPath()메소드를 통해 경로를 바꿔서 그려주고 removeHiddenSpot(hiddenHSpot) 메소드로 발견한 위험지점을 숨겨진 위험지점 리스트에서 제거한다. 그리고 commandMovementAndChangePosInfo(hazardFound) 메소드로 움직임을 지시하고 끝날경우 위치,방향정보를 받아 수정하고 로봇의 위치를 바꿔서 그려준다. 모든 object spot 을 다 방문했을경우 메시지를 띄워준다. Return to menu 버튼을 누르면 메인으로 돌아간다.

1. setInitialInfo() 메소드는 mapData 로부터 각종 지점들 및 로봇의 위치, 맵의 크기에 대한 정보를 받아온다.
2. generateRondomSpot(self, hazardNum, cbNum) 메소드는 MapData 클래스의 get HazardSpot() 메소드와 getObjectSpot(), getStartSpot() 메소드를 통해 좌표들을 받고, 이 좌표들을 제외한 무작위 좌표에 hazard spot 과 color blob spot 을 각각 hazardNum, cbNum 개 만큼 생성한다. hazardNum 이나 cnNum 이 음수일 경우 0~(맵의 가로 길이 + 맵의 세로 길이) / 2 개 사이로 생성한다. 이때, 랜덤 생성한 hazard spot 이 object spot 을 전후좌우로 감싸고 있는 경우 그들 중 무작위로 하나를 골라 삭제한다. 최종 hazard spot 과 color blob spot 을 MapData 객체의 setHiddenData(hazard, cb) 메소드를 통해 해당 객체에 저장한다.
3. createPath()는 초기경로를 만드는 메소드이며 changePath()는 위험지점 발견으로인해 경로를 바꿔줘야할 때 사용하고 drawPath() 메소드는 이러한 경로를 그려준다.
4. controlRobotMovement() 메소드와 commandMovementAndChangePosInfo(hazardFound) 메소드는 로봇을 이동시키고 결과를 맵에 반영한다.
5. 숨겨진 위험지점을 발견할 경우 숨겨진 위험지점 리스트에서 발견한 곳을 제거하기위해 removeHiddenSpot(hiddenHSpot) 메소드를 사용한다.

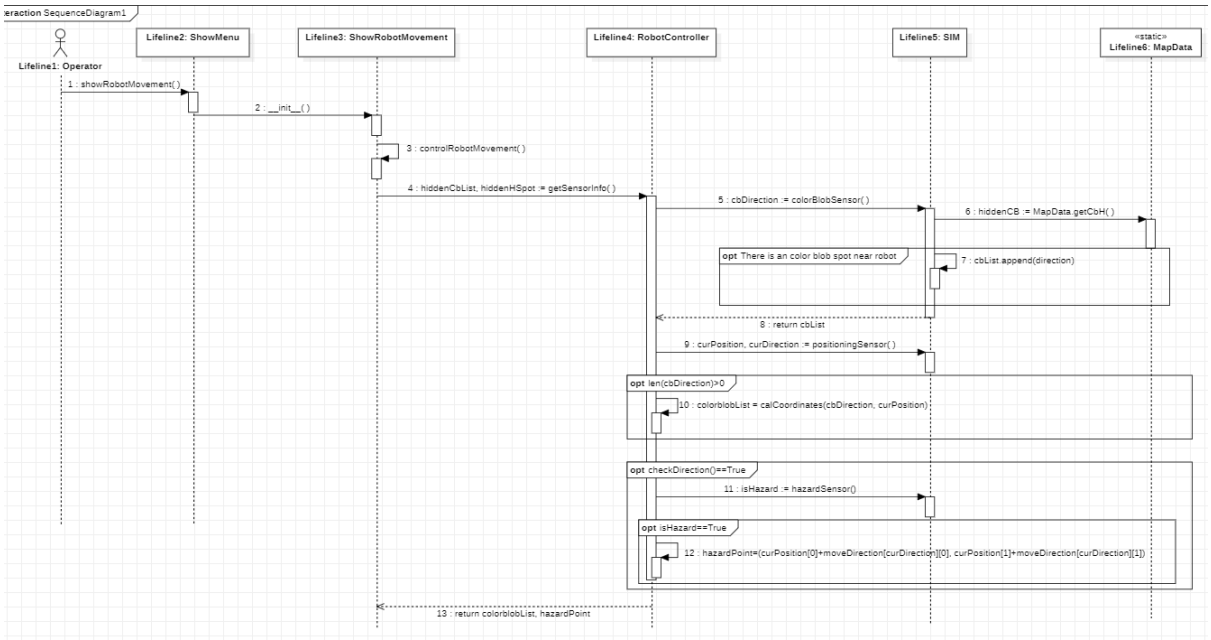
4.4 Use Case : Command robot movement



CommandMovement() 메소드는 현재 만들어져 있는 경로와 로봇의 센서를 통해서 얻은 로봇의 현재 위치, 현재 방향을 확인하고 mobile robot 에게 그에 맞는 동작을 지시한다. 지시하는 동작은 moveForward() 메소드로 앞으로 1 칸 이동, turnCW() 메소드로 시계 방향으로 90 도 회전 중 하나이다. 앞으로 1 칸 이동을 지시했을 때는 positioningSensor() 메소드로 다시 한 번 로봇의 위치를 확인한 뒤 두 칸 앞으로 이동했으면 drawRobot(changedPosition, changedDirection) 메소드로 새로운 경로를 만든다. 이 Use Case 는 'Show robot movement to all object spot' Use Case 에 포함된다.

1. moveForward() 메소드는 로봇을 한 칸 앞으로 움직이게 한다.
2. turnCw() 메소드는 로봇의 현재 방향이 경로 상에서 앞으로 나아가야 할 방향과 맞지 않을 때, 로봇을 시계 방향으로 90 도 회전하게 한다.
3. moveForward() 메소드를 실행하면 SIM 클래스의 positioningSensor() 메소드를 통해 로봇의 현재 위치를 확인하고, 앞으로 두 칸 이동했으면 drawRobot(changedPosition, changedDirection) 메소드로 새로운 경로를 만든다.

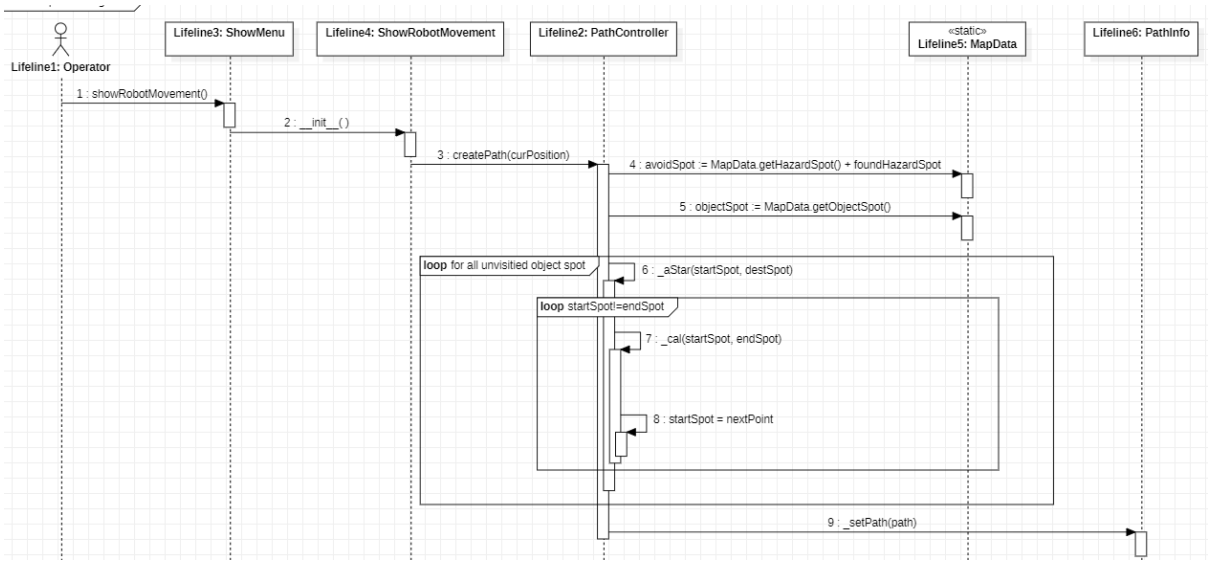
4.5 Use Case : Check spot near the robot



GetSensorInfo() 메소드는 현재 mobile robot 의 주변에 존재하는 hazard spot 과 color blob spot 의 좌표를 받는다. 해당 정보들은 mobile robot 의 센서들을 이용하여 RobotController 클래스 내부에서 계산된다. hazard spot 을 발견하면 MapData 객체에 저장되어 있는 해당 좌표를 삭제한다.

1. positioningSensor() 메소드는 mobile robot 에게서 현재 로봇의 위치와 보고 있는 방향을 받는다.
2. hazardSensor() 메소드는 mobile robot 의 hazard sensor 를 통해 현재 로봇 전방 1 칸에 hazard spot 이 있는지 확인한다.
3. colorBlobSensor() 메소드는 mobile robot 의 color blob sensor 를 통해 현재 로봇의 위치 전후좌우에 color blob spot 이 있는지 확인한다.

4.6 Use Case : Create robot path to all unvisited obeject spot

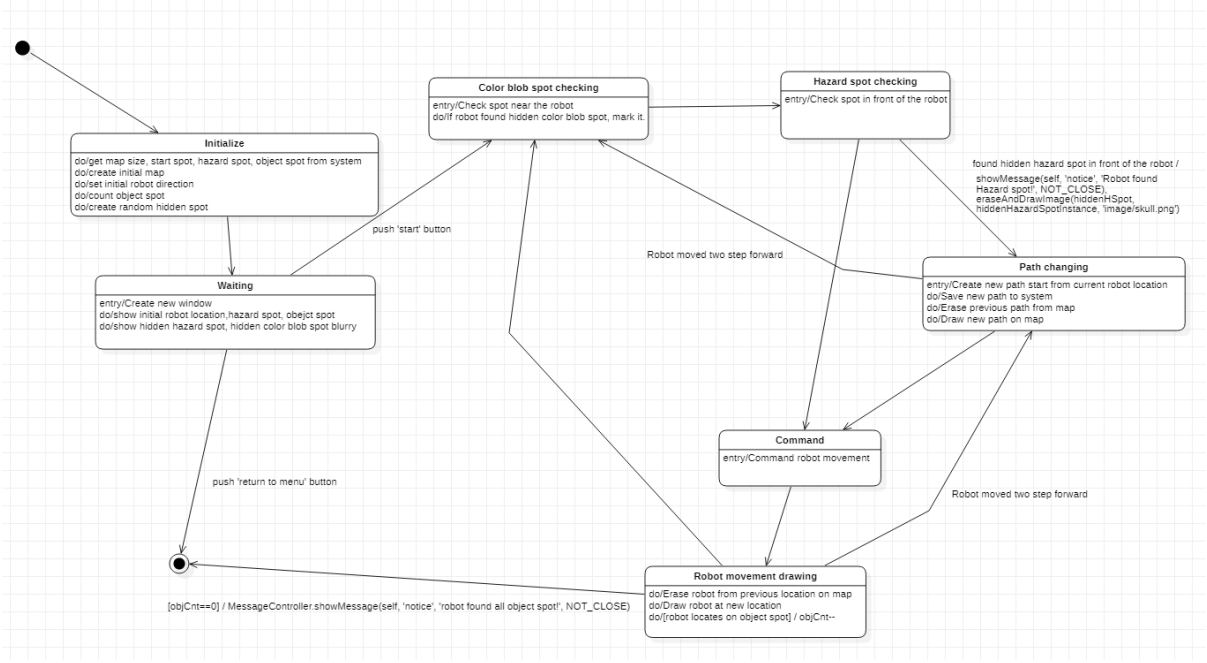


현재 위치정보를 받아서 createPath(curPosition) 메소드로 경로를 만든다. 경로를 만들 때 발견되거나 알고 있는 위험지점을 avoidSpot 으로 지정하고 중요지점을 objectSpot 으로 지정해서 경로를 만든다. 경로 제작 알고리즘으로는 A*알고리즘을 사용하며 경로가 완성되면 PathInfo 객체에 저장한다.

1. createPath(curPosition) 메소드는 전달받은 현재 위치와 getHazardSpot(), FoundHazardSpot(), getObjectSpot() 메소드로 얻은 지점을 바탕으로 경로를 만든다.
2. _aStar(self, start, end) 메소드는 start 지점에서 end 지점까지 가는 경로를 구하는 a* 알고리즘을 수행한다. 세 개의 2 차원 리스트 close list, open list, parent 를 관리한다. close list 는 이미 방문했거나 hazard spot 인 지점을 넣는다. open list 는 현재 지점 상하좌우 지점에서 close list 에 들어가지 않는 지점들을 넣는다. parent 는 목적지에 도착했을 때 parent 배열을 역순으로 탐색하여 만들어진 경로를 찾기 위해 사용한다. 현재 지점에서 다른 지점으로 이동할 때 그 지점의 parent 배열의 값을 현재 지점으로 설정해준다.

5 객체 상태도

5.1 ShowRobotMovement 객체 상태도



ShowRobotMovement 는 Operator 에게 mobile robot 의 이동 과정을 보여주는 클래스이다. 해당 클래스는 다음과 같은 일곱 가지의 상태 중 하나를 가진다.

1. Initialize

   Operator 가 메뉴에서 Show robot movement 항목을 선택했을 때 시스템은 새로운 ShowRobotMovement 객체를 만든다. MapData 객체의 정보를 바탕으로 초기 맵을 만들고 로봇의 초기 위치와 방향을 설정한다. 그리고 목표지점을 세어서 objCnt 에 넣는다.

2. Waiting

   초기 설정단계가 끝나면 새 창을 띄워 초기 맵 상태와 함께 로봇의 현재 위치, 동작, 남은 목표지점 개수를 알려주는 groupbox 와 맵 상의 아이콘이 어떤 의미를 가지는지 알려주는 표를 Operator 에게 보여준다.

3. Color blob spot checking

   Waiting 상태에서 start 버튼을 누르면 이 상태로 이동한다. 로봇 주변에 hidden color blob spot 이 있는지 확인한다. 로봇 근처에서 발견한 hidden color blob 들을 맵에 표시한다. 이 작업이 끝나면 Hazard spot checking 상태로 이동한다.

4. Hazard spot checking

   로봇 바로 앞에 hidden hazard spot 이 있는지 확인한다. 있으면 메시지를 띄운 뒤 Path changing 상태로, 없으면 Command 상태로 전이한다.

5. Path changing

   Hazard spot checking 상태에서 hidden hazard spot 을 발견했거나, 로봇이 오작동으로 인해 두 칸 앞으로 이동했을 때 이 상태로 이동한다. Hazard spot 을 피해서 현재 로봇의 위치로부터 새로운 경로를 만들고 시스템에 저장한다. 이전의 경로를 지우고 새로 만든 경로를 맵에 그려준다. Hidden hazard spot 을 발견해 이 상태로 들어왔으면 Command 상태로, 오작동으로 인해 이 상태로 들어왔으면 Color blob spot checking 상태로 전이한다.

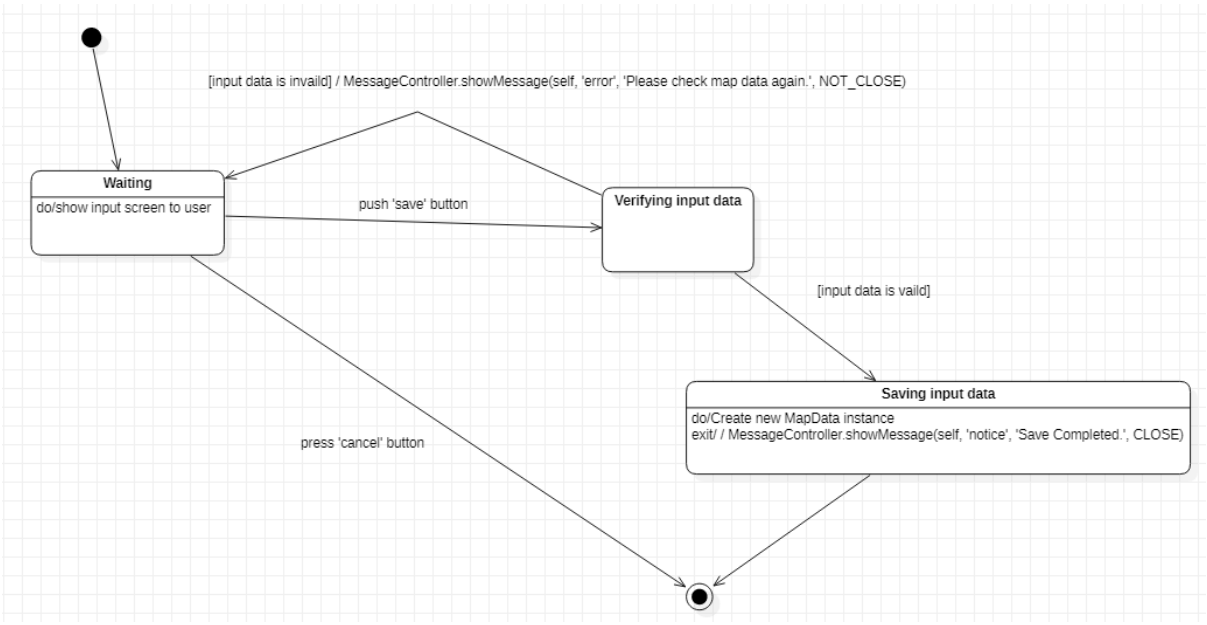
6. Command

   Hazard spot checking 상태에서 로봇 앞에 숨겨진 hazard spot 이 없거나 Path changing 상태에서 새 경로를 만들고 이를 맵에 그리고 난 뒤 이 상태로 이동한다. 로봇의 이동을 명령한다.

7. Robot movement drawing

   Command 상태에서 로봇의 이동을 명령받았을 때 이 상태로 이동한다. 이전의 로봇의 위치를 지우고 새로운 위치에 로봇을 다시 그려준다. Object spot 에 도착한 경우 objCnt 를 하나 줄여준다. 이때 objCnt 가 0 이 될 경우 “robot found all object spot!” 메시지를 띄우고 그렇지 않을 경우 다시 Color blob spot checking 상태로 돌아간다.

5.2 SaveData 객체 상태도



saveData 객체는 사용자가 입력한 맵 정보들을 저장한다. saveData 객체는 다음과 같은 세 가지 상태 중 하나를 가진다.

1. Waiting

   처음 saveData 객체를 생성하면 이 상태에 진입한다. 사용자에게 맵 정보를 입력받는다. 사용자가 save 버튼을 누르거나 cancel 버튼을 누르면 다른 상태로 전이된다.

2. Verifying input data

   Waiting 상태에서 Save 버튼을 눌렀을 경우 이 상태에 진입한다. 데이터가 유효한 값을 가지는지 확인한다. 유효하지 않을 경우 please check map data again 메시지를 출력하고 Waiting 상태로 돌아간다.

3. Saving input data

   Verifying input data 에서 데이터가 유효할 경우 이 상태에 진입한다. MapData 클래스를 통해 입력 값들을 저장한 뒤 "Save Completed." 메시지를 출력하고 메뉴화면으로 돌아간다.