객체지향개발방법론

TRANDITONAL VS VIBE

4팀 김승찬, 유호재, 윤현서, 최서진

INDEX

목차페이지

UC - 03. 고립탈출 (Brief)

[변경]

주행 중 전방, 좌측, 우측 센서로부터 장애물이 감지된다.

→ 주행 중 전방, 좌측, 우측에서 장애물이 감지된다.

Traditional - Inception

FR - 04. 우선순위 회피 주행

[변경]

전방 장애물 발생 시 [우측 → 좌측]

→ [좌측 → 우측] 순으로 가용 공간을 확인하여 방향을 전환해야 한다.

Traditional - Inception

FR - 05. 복합 장애물 대응

[변경]

전방, 좌, 우가 모두 차단된 경우 후진 후 우측으로 방향을 전환하여
→ 회전 방향을 판단하여 주행을 재개해야 한다.

UC - 03. 고립탈출 (Casual)

[삭제]

1. (A) Obstacle Sensor이 System에 전방, 좌측, 우측에 장애물이 있음을 의미하는 센서 데이터를 알린다.

[추가]

2. (S) System 이 우측 회전과 전방 센서 데이터로 우측 장애물을 감지한다.

Traditional - OOD

UC - 01. 직선 청소 주행 (Fully Dressed)

[변경]

4. (S) System은 회전 가능한 방향을 판단하고 Motor를 제어하여 방향을 전환한다

→ '->UC 2. 장애물 회피' 시나리오로 전환한다.

UC - 02. 고립 탈출 (Fully Dressed)

[추가]
우측으로 3초간 90도 회전한 뒤 전방 센서로 우측 장애물 여부를 확인한다.

[추가]
4. (S) System은 우측 장애물 스캔을 위해 우측으로 3초간 90도 회전한다.

[변경]
1.(O) Obstacle Sensor에서 System에 좌측, 우측 역시 장애물이 있음을 의미하는 센서 데이터를 알린다

↓

1. (S) System은 좌측 센서 데이터와 우측 90도 스캔 결과를 바탕으로 좌측, 우측 역시 장애물이 있음을 판단한다.

UC - 03. 고립 탈출 (Fully Dressed)

[변경]

전방, 좌측, 우측에서 obstacle sensor 장애물 감지

→ 전방, 좌측 장애물과 우측 90도 스캔 결과로 우측 장애물이 판단됨

[삭제]

1. (O) Obstacle Sensor이 System에 전방, 좌측, 우측에 장애물이 있음을 의미하는 센서 데이터를 알린다.

[추가]

2. (S) System은 우측으로 3초간 90도 회전한 뒤 전방 센서로 우측 장애물 여부를 판단한다.

Traditional - OOD

UC - 03. 고립 탈출 (Fully Dressed)

[변경]

1. (O)Obstacle Sensor에서 System에 우측, 좌측에 장애물이 있음을 의미하는 센서 데이터를 알린다.

↓

1. (S) System은 좌측 센서 데이터와 우측 90도 스캔 결과를 바탕으로 좌측, 우측 모두에 장애물이 있음을 판단한다.

[변경]

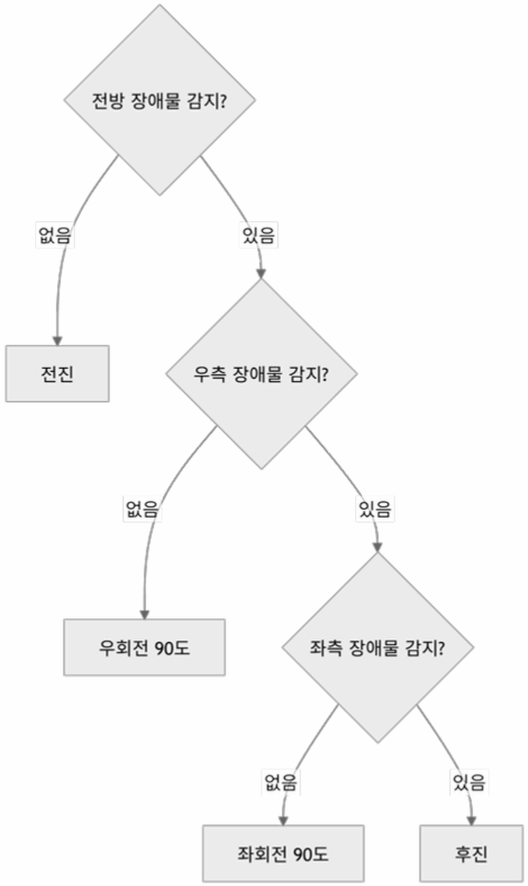
3. (O) Obstacle Sensor에서 양측에 장애물이 있는지 확인한다.

↓

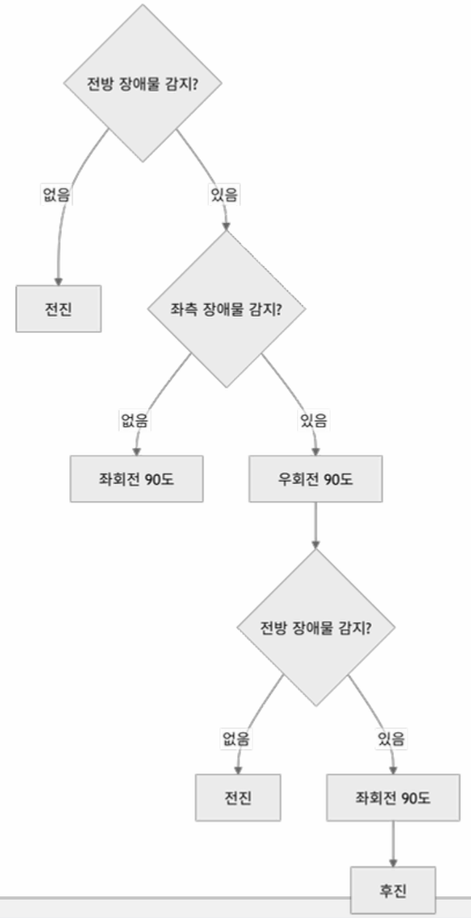
3. (S) System은 좌측 장애물과 우측 90도 스캔으로 확인한 우측 장애물이 있는지 판단한다.

Traditional-OOD

기존 로직



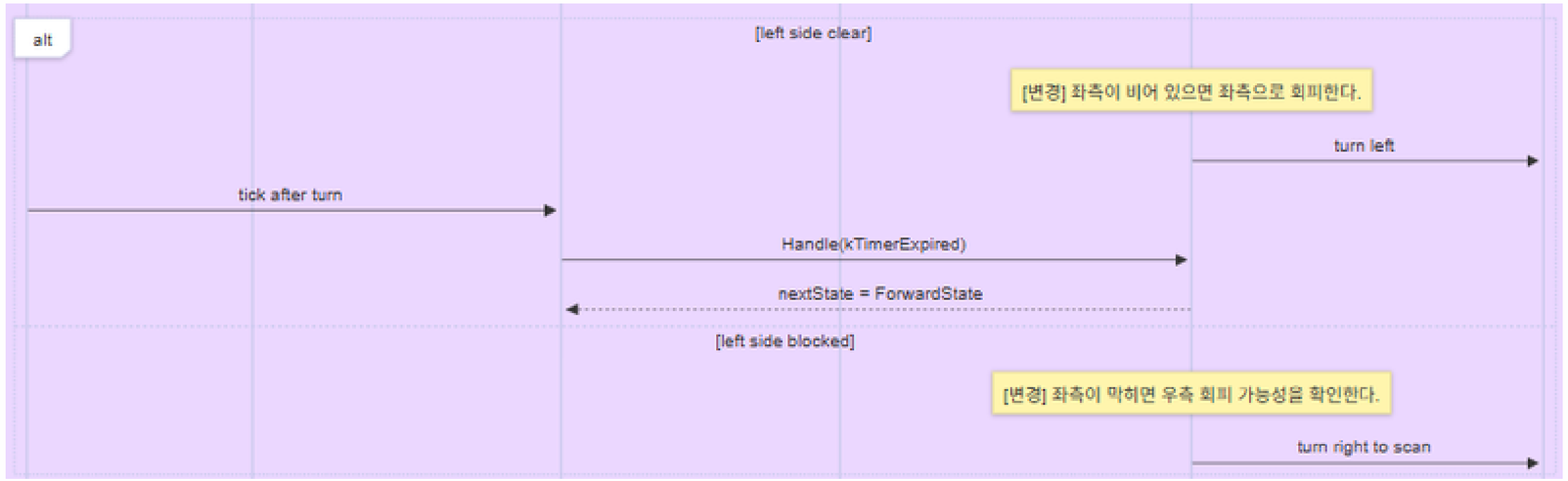
변경된 로직



Traditional-OOD

SD - 02 / SD - 03 공통

회피 시 좌회전 우선



Traditional-OOD

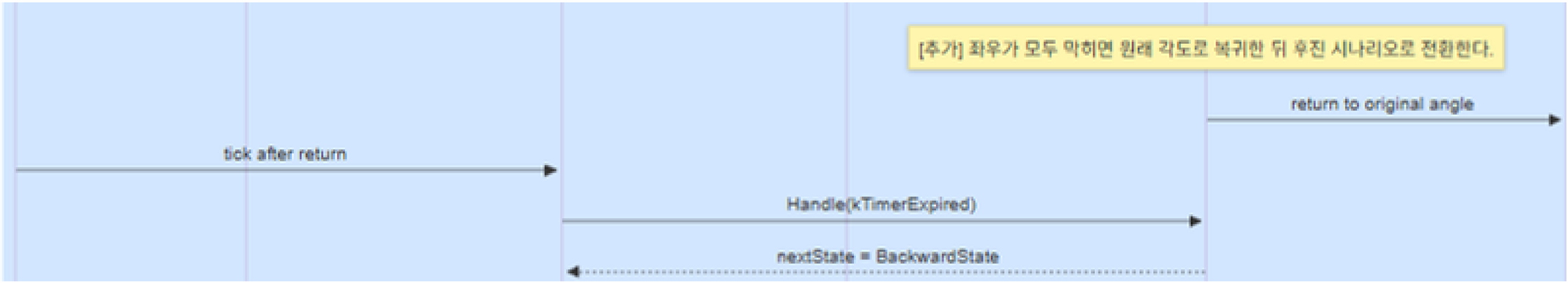
SD - 02 / SD - 03 공통

좌회전 불가능하면
우회전 → 우측 장애물 감지

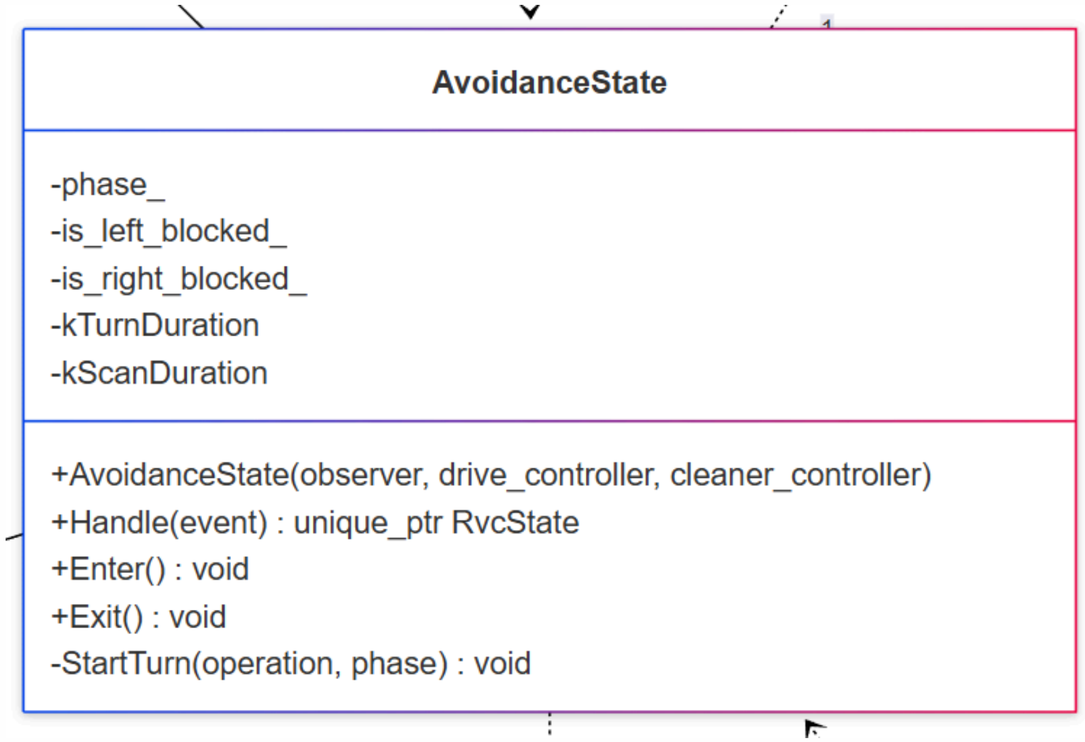
Traditional-OOD

SD - 02 / SD - 03 공통

우회전 불가능하면
좌회전 → 후진



Traditional-OOD



[추가]

phase_

내부 phase
(우측 장애물 확인을 위한 회전등)

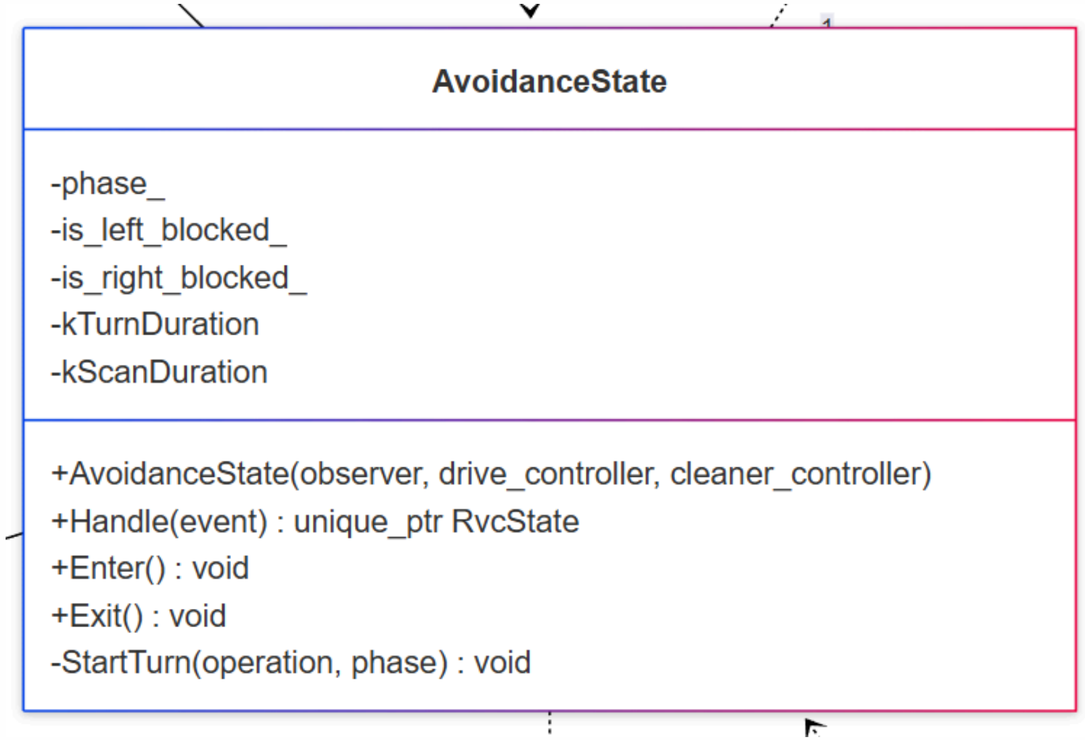
kTurnDuration

회전 시간

kScanDuration

장애물 감지 tick 수

Traditional-OOD



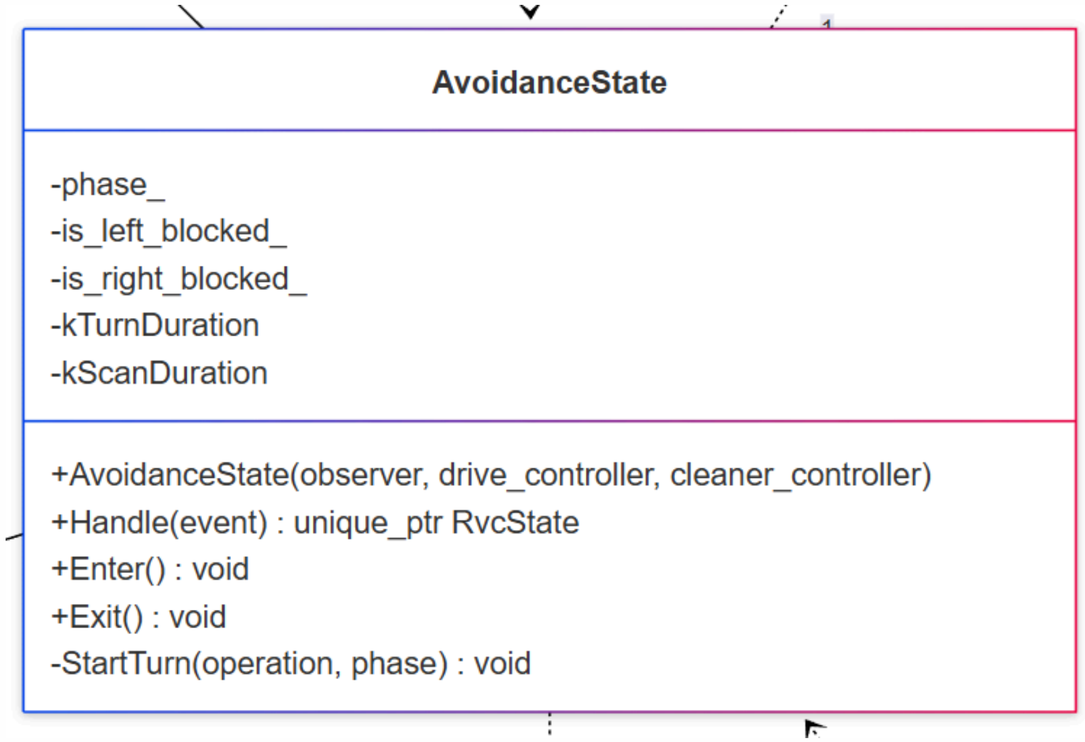
[삭제]

is_turning

현재 회전 상태 나타내는 플래그

내부 phase 도입으로 상태가 다양해지며 삭제

Traditional-OOD



[변경]

StartTurn()

결정된 회전 방향으로 실제 회전 지시

내부 phase 를 인자로 전달
phase 마다 회전의 목적 다름

회전 목적

- **장애물 회피**
- **장애물 탐색**

Traditional-OOI

```
[----------] 5 tests from SensorInterfaceTest
[ RUN      ] SensorInterfaceTest.ReadSensorSuccess
[       OK ] SensorInterfaceTest.ReadSensorSuccess (0 ms)
[ RUN      ] SensorInterfaceTest.ReadSensorFailWhenPartialSize
[       OK ] SensorInterfaceTest.ReadSensorFailWhenPartialSize (0 ms)
[ RUN      ] SensorInterfaceTest.ReadSensorFailWhenPartialRead
[       OK ] SensorInterfaceTest.ReadSensorFailWhenPartialRead (0 ms)
[ RUN      ] SensorInterfaceTest.ReadSensorFailWhenOverSize
[       OK ] SensorInterfaceTest.ReadSensorFailWhenOverSize (0 ms)
[ RUN      ] SensorInterfaceTest.ReadSensorFailWhenFstatFails
[       OK ] SensorInterfaceTest.ReadSensorFailWhenFstatFails (0 ms)
[----------] 5 tests from SensorInterfaceTest (1 ms total)
```

```
[----------] 2 tests from CleanerControllerTest
[ RUN      ] CleanerControllerTest.SetHappyPath
[       OK ] CleanerControllerTest.SetHappyPath (0 ms)
[ RUN      ] CleanerControllerTest.SetSadPath
[       OK ] CleanerControllerTest.SetSadPath (0 ms)
[----------] 2 tests from CleanerControllerTest (0 ms total)
```

```
[----------] 5 tests from SensorInterfaceTest
[ RUN      ] SensorInterfaceTest.ReadSensorSuccess
[       OK ] SensorInterfaceTest.ReadSensorSuccess (0 ms)
[ RUN      ] SensorInterfaceTest.ReadSensorFailWhenPartialSize
[       OK ] SensorInterfaceTest.ReadSensorFailWhenPartialSize (0 ms)
[ RUN      ] SensorInterfaceTest.ReadSensorFailWhenPartialRead
[       OK ] SensorInterfaceTest.ReadSensorFailWhenPartialRead (0 ms)
[ RUN      ] SensorInterfaceTest.ReadSensorFailWhenOverSize
[       OK ] SensorInterfaceTest.ReadSensorFailWhenOverSize (0 ms)
[ RUN      ] SensorInterfaceTest.ReadSensorFailWhenFstatFails
[       OK ] SensorInterfaceTest.ReadSensorFailWhenFstatFails (0 ms)
[----------] 5 tests from SensorInterfaceTest (1 ms total)
```

```
[----------] 7 tests from DustSensorControllerTest
[ RUN      ] DustSensorControllerTest.NotifiesHighDustAtIntMax
[       OK ] DustSensorControllerTest.NotifiesHighDustAtIntMax (1 ms)
[ RUN      ] DustSensorControllerTest.DoesNotNotifyWhenAt0
[       OK ] DustSensorControllerTest.DoesNotNotifyWhenAt0 (1 ms)
[ RUN      ] DustSensorControllerTest.DoesNotNotifyWhenLowerThanThreshold
[       OK ] DustSensorControllerTest.DoesNotNotifyWhenLowerThanThreshold (0 ms)
[ RUN      ] DustSensorControllerTest.DoesNotNotifyWhenAtThreshold
[       OK ] DustSensorControllerTest.DoesNotNotifyWhenAtThreshold (0 ms)
[ RUN      ] DustSensorControllerTest.NotifiesHighDustWhenAboveThreshold
[       OK ] DustSensorControllerTest.NotifiesHighDustWhenAboveThreshold (0 ms)
[ RUN      ] DustSensorControllerTest.NotifiesFaultOnReadFailure
[       OK ] DustSensorControllerTest.NotifiesFaultOnReadFailure (0 ms)
[ RUN      ] DustSensorControllerTest.NotifiesFaultOnNegativeValues
[       OK ] DustSensorControllerTest.NotifiesFaultOnNegativeValues (0 ms)
[----------] 7 tests from DustSensorControllerTest (7 ms total)
```

Traditional-OOI

```
[----------] 8 tests from ActuatorInterfaceTest
[ RUN      ] ActuatorInterfaceTest.PreadFail
[       OK ] ActuatorInterfaceTest.PreadFail (0 ms)
[ RUN      ] ActuatorInterfaceTest.RawStatusNotOne
[       OK ] ActuatorInterfaceTest.RawStatusNotOne (0 ms)
[ RUN      ] ActuatorInterfaceTest.HappyPath
[       OK ] ActuatorInterfaceTest.HappyPath (0 ms)
[ RUN      ] ActuatorInterfaceTest.BadStatusAndReturnBad
[       OK ] ActuatorInterfaceTest.BadStatusAndReturnBad (0 ms)
[ RUN      ] ActuatorInterfaceTest.HappyMotor
[       OK ] ActuatorInterfaceTest.HappyMotor (0 ms)
[ RUN      ] ActuatorInterfaceTest.HappyInt
[       OK ] ActuatorInterfaceTest.HappyInt (0 ms)
[ RUN      ] ActuatorInterfaceTest.HappyCleaner
[       OK ] ActuatorInterfaceTest.HappyCleaner (0 ms)
[ RUN      ] ActuatorInterfaceTest.1stFails_2ndFine
[       OK ] ActuatorInterfaceTest.1stFails_2ndFine (0 ms)
[----------] 8 tests from ActuatorInterfaceTest (0 ms total)
```

```
[----------] 5 tests from DriveControllerTest
[ RUN      ] DriveControllerTest.HappyPathDriving
[       OK ] DriveControllerTest.HappyPathDriving (0 ms)
[ RUN      ] DriveControllerTest.HappyPathLeft
[       OK ] DriveControllerTest.HappyPathLeft (0 ms)
[ RUN      ] DriveControllerTest.HappyPathStop
[       OK ] DriveControllerTest.HappyPathStop (0 ms)
[ RUN      ] DriveControllerTest.SadPathRight
[       OK ] DriveControllerTest.SadPathRight (0 ms)
[ RUN      ] DriveControllerTest.SadPathBoth
[       OK ] DriveControllerTest.SadPathBoth (0 ms)
[----------] 5 tests from DriveControllerTest (0 ms total)
```

```
[----------] 5 tests from RvcSystemTest
[ RUN      ] RvcSystemTest.TickCallsSensorsAndState
[       OK ] RvcSystemTest.TickCallsSensorsAndState (0 ms)
[ RUN      ] RvcSystemTest.MultipleTickCalls
[       OK ] RvcSystemTest.MultipleTickCalls (0 ms)
[ RUN      ] RvcSystemTest.NoStateTransitionWhenHandleReturnsNull
[       OK ] RvcSystemTest.NoStateTransitionWhenHandleReturnsNull (0 ms)
[ RUN      ] RvcSystemTest.NotifyTriggerStateTransition
[       OK ] RvcSystemTest.NotifyTriggerStateTransition (0 ms)
[ RUN      ] RvcSystemTest.NotifyThrowsExceptionOnShutdown
[       OK ] RvcSystemTest.NotifyThrowsExceptionOnShutdown (0 ms)
[----------] 5 tests from RvcSystemTest (1 ms total)
```

```
[----------] 8 tests from AvoidanceStateTest
[ RUN      ] AvoidanceStateTest.ExitAndStopDriving
[       OK ] AvoidanceStateTest.ExitAndStopDriving (0 ms)
[ RUN      ] AvoidanceStateTest.LeftClearStartsLeftTurnAndSwitchesToForward
[       OK ] AvoidanceStateTest.LeftClearStartsLeftTurnAndSwitchesToForward (0 ms)
[ RUN      ] AvoidanceStateTest.LeftBlockedStartsRightScanTurn
[       OK ] AvoidanceStateTest.LeftBlockedStartsRightScanTurn (0 ms)
[ RUN      ] AvoidanceStateTest.RightClearAfterRightScanSwitchesToForward
[       OK ] AvoidanceStateTest.RightClearAfterRightScanSwitchesToForward (0 ms)
[ RUN      ] AvoidanceStateTest.BothSidesBlockedReturnsThenSwitchesToBackward
[       OK ] AvoidanceStateTest.BothSidesBlockedReturnsThenSwitchesToBackward (0 ms)
[ RUN      ] AvoidanceStateTest.LeftCheckWindowLastsOneTick
[       OK ] AvoidanceStateTest.LeftCheckWindowLastsOneTick (0 ms)
[ RUN      ] AvoidanceStateTest.IgnoreUnrelatedEvents
[       OK ] AvoidanceStateTest.IgnoreUnrelatedEvents (0 ms)
[ RUN      ] AvoidanceStateTest.SadPathLMotorFail
[       OK ] AvoidanceStateTest.SadPathLMotorFail (0 ms)
[----------] 8 tests from AvoidanceStateTest (0 ms total)
```

Traditional-OOI

```
[----------] 4 tests from ErrorStateTest
[ RUN      ] ErrorStateTest.HappyPathStopHW
[System] Emergency Stop: All actuators powered down.
**************************************
*          FATAL SYSTEM ERROR          *
*       RECOVERY IS NOT POSSIBLE       *
*    SYSTEM WILL SHUTDOWN SHORTLY      *
**************************************
[LOG] Writing fatal error log to persistent storage...
[LOG] Error Code: 0xffffff01
[LOG] State: ErrorState transition complete.
[       OK ] ErrorStateTest.HappyPathStopHW (0 ms)
[ RUN      ] ErrorStateTest.HappyPathSafelyExit
[       OK ] ErrorStateTest.HappyPathSafelyExit (0 ms)
[ RUN      ] ErrorStateTest.HappyPath5secShutdown
[       OK ] ErrorStateTest.HappyPath5secShutdown (0 ms)
[ RUN      ] ErrorStateTest.IgnoreUnrelatedEvents
[       OK ] ErrorStateTest.IgnoreUnrelatedEvents (0 ms)
[----------] 4 tests from ErrorStateTest (0 ms total)
```

```
[----------] 6 tests from ForwardStateTest
[ RUN      ] ForwardStateTest.HappyPathHWActivate
[       OK ] ForwardStateTest.HappyPathHWActivate (0 ms)
[ RUN      ] ForwardStateTest.ExitStopsDriveAndCleaning
[       OK ] ForwardStateTest.ExitStopsDriveAndCleaning (0 ms)
[ RUN      ] ForwardStateTest.HappyPathSwitch2Avoidance
[       OK ] ForwardStateTest.HappyPathSwitch2Avoidance (0 ms)
[ RUN      ] ForwardStateTest.HappyPathPowerCleaning
[       OK ] ForwardStateTest.HappyPathPowerCleaning (0 ms)
[ RUN      ] ForwardStateTest.HapplyPathBack2Cleaning
[       OK ] ForwardStateTest.HapplyPathBack2Cleaning (0 ms)
[ RUN      ] ForwardStateTest.SadPathLMotorFail
[       OK ] ForwardStateTest.SadPathLMotorFail (0 ms)
[----------] 6 tests from ForwardStateTest (0 ms total)
```

```
[----------] 5 tests from BackwardStateTest
[ RUN      ] BackwardStateTest.HappyPathBackward
[       OK ] BackwardStateTest.HappyPathBackward (0 ms)
[ RUN      ] BackwardStateTest.HapplyPathExitAndStop
[       OK ] BackwardStateTest.HapplyPathExitAndStop (0 ms)
[ RUN      ] BackwardStateTest.HappyPathBackToAvoid
[       OK ] BackwardStateTest.HappyPathBackToAvoid (0 ms)
[ RUN      ] BackwardStateTest.SadPathLMotorFail
[       OK ] BackwardStateTest.SadPathLMotorFail (0 ms)
[ RUN      ] BackwardStateTest.IgnoreUnrelatedEvents
[       OK ] BackwardStateTest.IgnoreUnrelatedEvents (0 ms)
[----------] 5 tests from BackwardStateTest (0 ms total)
```

Traditional-OOI

LCOV - code coverage report

			Coverage	Total	Hit
Current view:	top level	Lines:	97.1 %	239	232
Test:	coverage_filtered.info	Functions:	97.1 %	34	33
Test Date:	2026-06-03 12:49:00				

Directory	Line Coverage			Function Coverage		
	Rate	Total	Hit	Rate	Total	Hit
src/	100.0 %	5	5	50.0 %	2	1
src/actuator/	96.2 %	52	50	100.0 %	8	8
src/core/	89.7 %	29	26	100.0 %	3	3
src/sensor/	96.2 %	26	25	100.0 %	2	2
src/state/	99.2 %	127	126	100.0 %	19	19

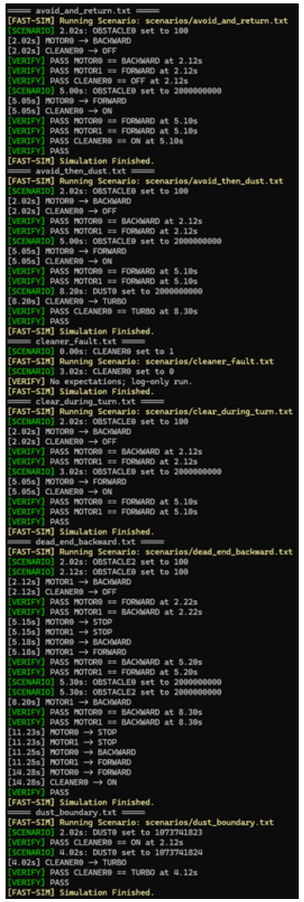
```
[info] Warm up run (threads: 1)
       [##################################] 1/1. Finished in 15ms
[info] Baseline run (threads: 1)
       [##################################] 1/1. Finished in 16ms
[info] Running mutants (threads: 28)
       [##################################] 30/30. Finished in 49ms
[info] Mutation score: 100%
[info] All mutations have been killed
[info] Total execution time: 214ms
```

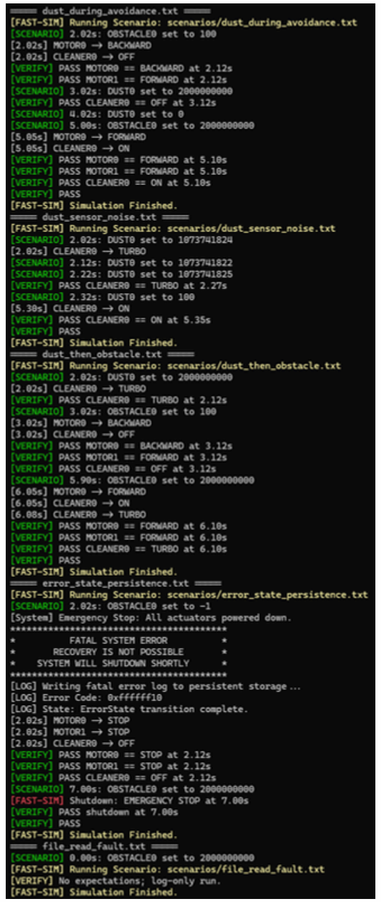
함수 Coverage 50%

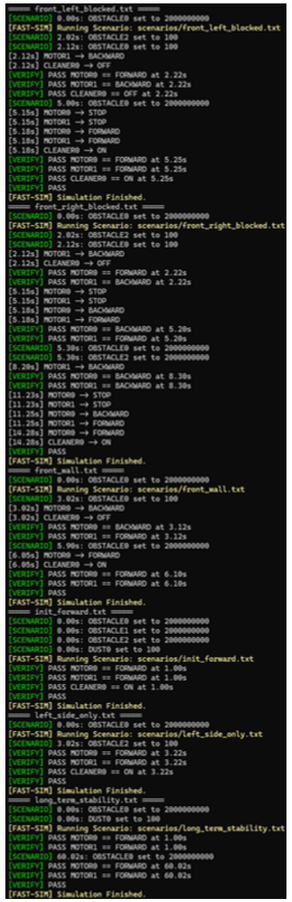
LCOV 가 가상 소멸자 하나를 2개로 인식

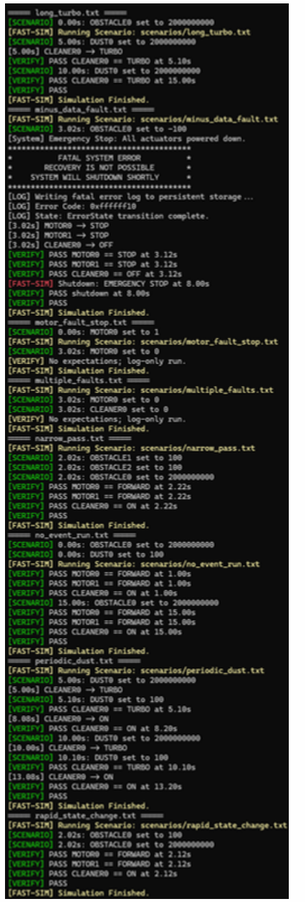
Line Coverage 기록에 의하면 정상

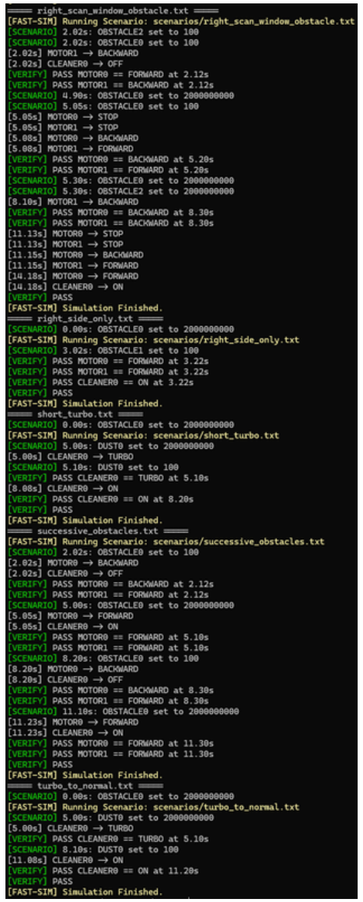
Traditional-OOI











VIBE-1차와 차이점 요약

1차 : 필요한 내용을 모두 prompt를 통해 넣어주는 vibe coding 수행

2차 : 커서에서 제공하는 기능(rules, skills, command)을 이용하여
하네스 엔지니어링과 유사한 환경 세팅.

prompt는 처음에 변동사항(우측 센서 제거) 제시, 최소 내용 + 체크하는 정도로 진행

VIBE-하네스 세팅

ooad에 필요한 절차, 비교 분석할 서류, CNC, 중요한 포인트 등 위주로 작성

SRS Editing

Instructions

When editing an SRS:

1. Ask the user what standard or requirement has changed and remember it.
2. Find the Functional Requirements (FR) and Non-Functional Requirements (NFR) related to the change in SRS file.
3. Update the FR and NFR so they meet the changed standard.
4. Check whether the FR and NFR are logically correct and achievable.
5. Review related SRS sections and update affected components.
6. Check completeness and consistency, especially names for objects, operations, actors, and requirements.
7. Preserve IEEE 830 template A.4 format. If the format is unclear, ask the user for the template.

usecase Editing

Instructions

When editing an usecase:

1. Look in new SRS file.
2. Find the updated parts of Functional Requirements (FR) and Non-Functional Requirements (NFR) and SRS
3. Review through the usecase file, focus on use case related to updated FR, NFR, and SRS.
4. Update the affected compoenets
5. Check completeness and consistency, especially names for actors, descriptions, course of events, and requirements within itself.
6. Check completeness and consistency, especially names for objects, operations, actors, descriptions, and requirements along with SRS.

특히, 빨간 상자처럼 확인할 서류와, CNC를 강조하며 작성

ssd Editing

Instructions

When editing an ssd:

1. Look in new usecase file.
2. Find the updated parts of usecase and its actors, descriptions, course of events, and requirements
3. update the components, actors, flow of logic of System Sequence Diagram.
4. keep the system part of diagram to be black boxed, the discription of things on the arrow must be operations with arguments.
5. keep the names of actors, arguments, operations human readable.
6. Check completeness and consistency, especially names for actors, operations, arguments and flow of logic within itself.
7. Check completeness and consistency, especially names for actors, operations, arguments and flow of logic with use case.
8. when everything is right, fix md file to have description of ssd, show the diagram via plantuml or mermaid. choose what it can see in md file. if you can't choose, ask user and tell him/her the pros and cons of each.

rules : 모르면 물어보라고 지시

Answer Style

- when there is a space for improvement in skill.md or the requested prompt, make 2 recommendations and 1 about how many major improvements can be made.
- Prefer to count major improvements without the 2 recommendations just made
- Prefer to explain briefly about major improvements like which file has what to improve
- Prefer direct recommendations over long lists of options.
- Ask a clarifying question before making assumptions that could cause rework.

이름

- classdiagram-editing
- domaindiagram-editing
- ooad-coding
- sd-editing
- simulator-coding
- srs-editing
- ssd-editing
- unittest-coding
- usecase-editing
- README.md

VIBE-하네스 세팅

domaian diagram Editing

Instructions

When editing an domain diagram:

1. Look in new ssd file.
2. Find the updated parts of its actors, flow of logic, opeation names and arguments
3. Look in new usecase file.
4. Find the updated parts of usecase and its actors, descriptions, course of events, and requirements
5. update the components of Domain Diagram.
6. keep the boxes to be business level of conceptual classes, not objects. also, draw the communication of those business level of conceptual classes.
7. the lines should show how many boxes can or are communicating with the connected box and write brief discription if necessary. for example, multiple airplane *---(flies to)---1 airport.
8. keep the names of box, arguments human readable and same as SSD.
9. Check completeness and consistency, especially names for business level conceptual class name, arguments and flow of logic within itself.
10. Check completeness and consistency, especially names for business level conceptual class name, arguments and flow of logic with SSD.
11. Check completeness and consistency, especially names for business level conceptual class name, arguments and flow of logic with usecase.
12. when everything is right, fix md file to have description of domain diagram, show the diagram via plantuml or mermaid. choose what it can see in md file. if you can't choose, ask user and tell him/her the pros and cons of each.

sd Editing

Instructions

When editing an sd:

1. Look in new usecase file.
2. Find the updated parts of usecase and its actors, descriptions, course of events, and requirements
3. update the components, actors, flow of logic of Sequence Diagram.
4. elaborate the system part of diagram with interaction with inner objects, and the discription of things on the arrow must be operations with arguments.
5. keep the names of actors, arguments, operations, object names human readable.
6. Check completeness and consistency, especially names for actors, object names, operations, arguments and flow of logic within itself.
7. Check completeness and consistency, especially names for actors, object names, operations, arguments and flow of logic with use case.
8. Check completeness and consistency, especially names for actors, object names, operations, arguments and flow of logic with ssd.
9. Check completeness and consistency, especially names for actors, object names, operations, arguments and flow of logic with SRS.
10. when everything is right, fix md file to have description of sd, show the diagram via plantuml or mermaid. choose what it can see in md file. if you can't choose, ask user and tell him/her the pros and cons of each.

VIBE-하네스 세팅

classdiagram Editing

Instructions

When editing an class diagram:

1. Look in new usecase file.
2. Find the updated parts of usecase and its actors, descriptions, course of events, and requirements
3. update the components, actors, flow of logic of Class Diagram.
4. Look in new SD file.
5. Find the updated parts of SD and its actors, object names, flow of logic, operation names, and argument names
6. update the actors, object names, flow of logic, operation names, and argument names of Class Diagram.
7. keep the names of actors, arguments, operations, object names human readable.
8. make sure class diagram shows interaction between objects using right association, inheritance, relization, dependency, aggregation, composition.
9. Check completeness and consistency, especially names for actors, object names, operations, arguments and flow of logic within itself.
10. Check completeness and consistency, especially names for actors, object names, operations, arguments and flow of logic with use case.
11. Check completeness and consistency, especially names for actors, object names, operations, arguments and flow of logic with sd.
12. Check completeness and consistency, especially names for actors, object names, operations, arguments and flow of logic with SRS.
13. when everything is right, fix md file to have description of class diagram, show the diagram via plantuml or mermaid. choose what it can see in md file. if you can't choose, ask user and tell him/her the pros and cons of each.

ooad code Editing

Instructions

When editing an code:

1. Look in new class diagram file.
2. Find the updated parts of class diagram and its actors, operations, objects, flow of logic, interaction between classes/objects, and requirements
3. Look in new SD file.
4. Find the updated parts of SD and its actors, object names, flow of logic, operation names, and argument names
5. update the components, objects, flow of logic of the code.
6. keep the names of actors, arguments, operations, object names human readable and same as class diagram, SD.
7. make sure code follows the logic of class diagram and SD.
8. Check completeness and consistency, especially object names, operations, arguments and flow of logic within itself.
9. Check completeness and consistency, especially object names, operations, arguments and flow of logic with class diagram.
10. Check completeness and consistency, especially object names, operations, arguments and flow of logic with sd.
11. Check completeness and consistency, especially names for actors, object names with SRS.

VIBE-하네스 세팅

unit test Editing

Instructions

When editing an unit test:

1. Look in new code file.
2. Find the updated parts of code and its operations.
3. update the related unit test so it can test the updated code.
4. run code coverage
5. if the total code coverage is under 75%, look for issues and fix
6. when everything is done, fix md file to have description of each unit test and code coverage.

simulator Editing

Instructions

When editing an simulator:

1. Look in new code file.
2. Find the updated parts of code and its operations.
3. update the related simulator parts so it can test the updated code.
4. check if simulator runs 30 tests.
5. check which test is positive and negative.
6. check if test description is explaining properly
7. run the simulator and see if there is fails. if so, check test and code to see what is problem.
8. if the code is problem, fix the code to pass the test. if the test is problem, fix the test. repeat 7 and 8 until all test passes
9. if the problem doesn't get solved within 3 tries, alert user for further instructions.
10. when everything is done, fix md file to have description of each simulator test about what is positive or negative test and what it tests.

VIBE-변동사항 요약

우측 센서의 부재

->

우측으로 잠시 회전(probe)하여,
다른 센서로 우측 장애물 탐지

VIBE-SRS

SRS에는 7가지의 변경사항,
7가지 추가사항, 4가지의 삭제 사항 발생

변경 사항

- SRS는 더 이상 전용 우측 장애물 센서가 있다고 가정하지 않습니다.
- 우측 영역은 여전히 동작 모델(behavior model)의 일부이지만, 우측의 차단/개방(blocked/clear) 상태는 이제 로봇을 회전시켜 전면 장애물 센서로 우측을 검사하는 등 가용 센서를 활용하는 상위 수준의 우측 탐색(high-level right-side probe)을 통해 추론됩니다.
- '포위됨(Surrounded)'은 여전히 **전면**, **좌측**, **우측**이 모두 차단되었음을 의미합니다. 이 조건 중 우측 부분은 이제 우측 탐색 결과를 기반으로 합니다.
- '우회전 우선(Prefer-right) 회피'는 이제 "우측 탐색을 통해 우회전이 가능하다는 것이 확인된 후에만 우회전을 우선함"을 의미합니다.
- 통합 장애물 판정(Combined obstacle decisions)은 이제 전면/좌측의 직접적인 데이터와 우측 탐색 결과를 결합합니다.
- 구성(Configurability) 및 성능 요구 사항에 이제 우측 탐색의 타이밍과 파라미터가 포함됩니다.
- 탐색 자세(Probe-pose) 관측치는 별도의 우측 센서 메시지 대신 `ObstacleStateChanged(probePose=right, ...)`를 통해 표현됩니다.

추가 사항

- 우측 탐색(Right-side probe)에 대한 새로운 정의가 추가되었습니다.
- 전용 우측 센서 없이 우측 장애물 상태를 도출하기 위한 새로운 하드웨어 인터페이스 요구 사항 **HI-1A**가 추가되었습니다.
- 새로운 장애물 인식(obstacle-perception) 속성인 `rightObstacleInferred` 및 `rightProbeStatus`가 추가되었습니다.
- 우측 정보가 필요할 때 소프트웨어가 상위 수준의 탐색을 통해 우측 차단 여부를 결정하도록 요구하는 새로운 기능 요구 사항 **OBJ3-FR-5**가 추가되었습니다.
- 우측 탐색을 최종 회피/탈출 기동과 분리하는 새로운 내비게이션 요구 사항 **OBJ4-FR-8**이 추가되었습니다.
- 우측 탐색이 제한된 시간 내에 완료되거나 안전하게 대체 동작(fall back)을 수행하도록 요구하는 새로운 성능 요구 사항 **PERF-4**가 추가되었습니다.
- 하위(downstream) 설계, 코드, 테스트 및 시뮬레이터 산출물에 우측 센서의 부재를 반영하도록 요구하는 새로운 추적성(traceability) 요구 사항 **OTH-4**가 추가되었습니다.

삭제 / 제거 사항

- 소프트웨어 컨트롤러가 직접적인 우측 장애물 상태를 논리적 입력으로 사용해야 한다는 요구 사항이 제거되었습니다.
- 전면/좌측/우측 장애물 상태를 모두 센서에서 직접 얻는다는 가정이 제거되었습니다.
- "우측 센서 유효성(right sensor-valid)"을 직접적인 우측 센싱으로 취급하던 문구가 제거되었습니다. 이제 우측의 주행 가능 여부는 탐색(probe)을 통해 판단됩니다.
- 별도의 수신 메시지 이름인 `RightSideProbeResult`가 제거되었습니다. 탐색 관측치는 탐색 자세에서 발생한 `ObstacleStateChanged` 이벤트로 전달됩니다.

요약 : 우측 센서 제거, 우측은 회전을 통해 파악,
엄밀해진 정의, 관련 속성 추가/제거

VIBE-Usecase

UseCase에는 4가지의 변경사항,
3가지 추가사항, 2가지의 삭제 사항 발생

추가 사항

- 장애물 회피 및 포위 탈출 과정에 우측 탐색 단계를 추가했습니다.
- 탐색 실패 또는 시간 초과 시 UC-08 또는 제품 대체 동작(fallback)으로 이어지는 경로를 추가했습니다.
- HI-1A, OBJ3-FR-5, OBJ4-FR-8 및 PERF-4에 대한 현재 SRS v0.6.0의 추적성(traceability) 내용을 추가했습니다.

삭제 / 제거 사항

- 전면/좌측/우측 장애물 상태를 센서에서 직접 얻는다는 가정을 제거했습니다.
- 우회전 가능 여부를 우측 센서를 통해 직접 검증받는 것처럼 취급하던 문구를 제거했습니다.

변경 사항

- 직접적인 우측 장애물 센서라는 문구를 가용한 센싱을 활용하는 우측 탐색(right-side probe)으로 대체했습니다.
- 우측 방향에 대한 판단이 탐색 결과를 바탕으로 이루어지도록 UC-03, UC-04, UC-05, UC-08 및 UC-09를 업데이트했습니다.
- 포위됨(surrounded)의 의미를 '전면 + 좌측 + 우측 차단'으로 유지했으며, 우측 상태는 이제 추론을 통해 판단됩니다.
- 포위 탈출(surrounded escape) 시 필요할 경우 반드시 후진 이동을 사용해야 하며, 이를 180도 회전 후 전진하는 방식으로 대체해서는 안 된다는 점을 명확히 했습니다.

요약 : 우측 센서 관련 제거, 회전을 통한 우측 판단 내용 최신화, 엄밀해진 정의

VIBE-SSD

SDD에는 6가지의 변경사항, 4가지 추가사항, 2가지의 삭제 사항 발생

요약 : 우측 센서 관련 제거, 우측 확인하는 기능을 기존 함수에 추가

변경 사항

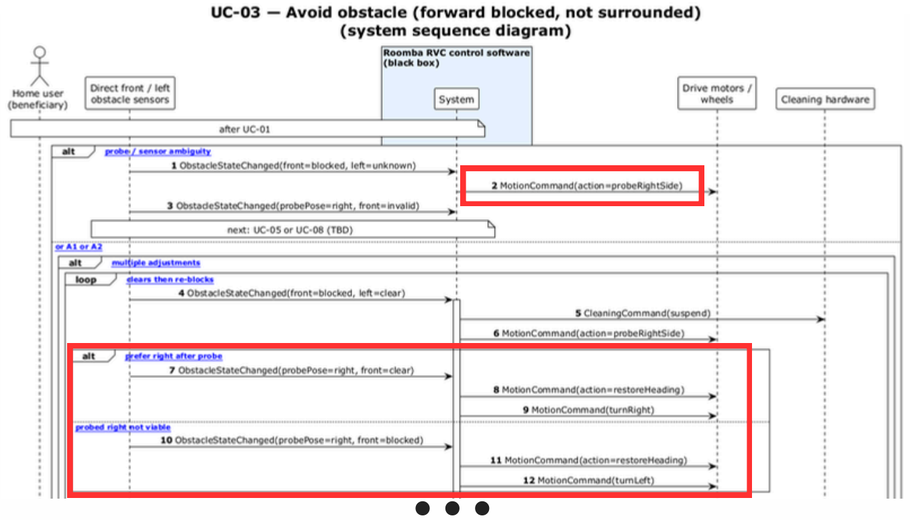
- UC-03, UC-04, UC-05, UC-08 및 UC-09 SSD(시스템 순서도)는 이제 전면/좌측/우측을 직접 센싱하는 대신 직접적인 **전면 / 좌측** 장애물 센서를 보여줍니다.
- 우측 방향에 대한 판단은 이제 상위 수준의 `MotionCommand(action=probeRightSide)` 및 `MotionCommand(action=restoreHeading)` 상호 작용을 통해 요청된 탐색 자세(probe pose)에서 얻은 직접적인 센서 관측치에서 도출되며, 이는 `ObstacleStateChanged(probePose=right, front=...)`로 표시됩니다.
- UC-05는 여전히 '포위됨(surrounded)'을 전면 + 좌측 + 우측 차단으로 간주하지만, 우측 상태는 이제 탐색(probe)을 통해 추론됩니다.
- UC-05는 이제 포위 탈출(surrounded escape) 시 후진 구간(reverse segment)이 필요하며, 180도 회전 후 전진하는 방식으로는 이를 대체할 수 없음을 명시적으로 기록합니다.
- UC-08은 이제 오래되거나(stale) 유효하지 않은 직접적인 장애물 데이터뿐만 아니라 우측 탐색 중 발생한 오래되거나 유효하지 않은 관측치도 다룹니다.
- UC-09는 이제 직접적인 전면/좌측 샘플과 탐색 자세 관측치를 결합하는 융합/스냅샷(fusion/snapshot) 로직을 보여줍니다.

추가 사항

- UC-03, UC-04, UC-05 및 UC-09에 우측 탐색 메시지가 추가되었습니다.
- UC-04 및 UC-08에 탐색 시간 초과(timeout) / 무효 경로(invalid path)가 추가되었습니다.
- 탐색 관련 프레임에 대한 마크다운 래퍼(wrapper) 설명이 추가되었습니다.
- SSD 래퍼 페이지와 소스 및 렌더링된 다이어그램 파일을 위한 정리된 로컬 링크가 추가되었습니다.

삭제 / 제거 사항

- 영향을 받는 SSD에서 직접적인 우측 장애물 센서에 대한 가정을 제거했습니다.
- 우회전 가능 여부를 직접적인 우측 센서 유효성과 동일하게 취급하던 문구를 제거했습니다.

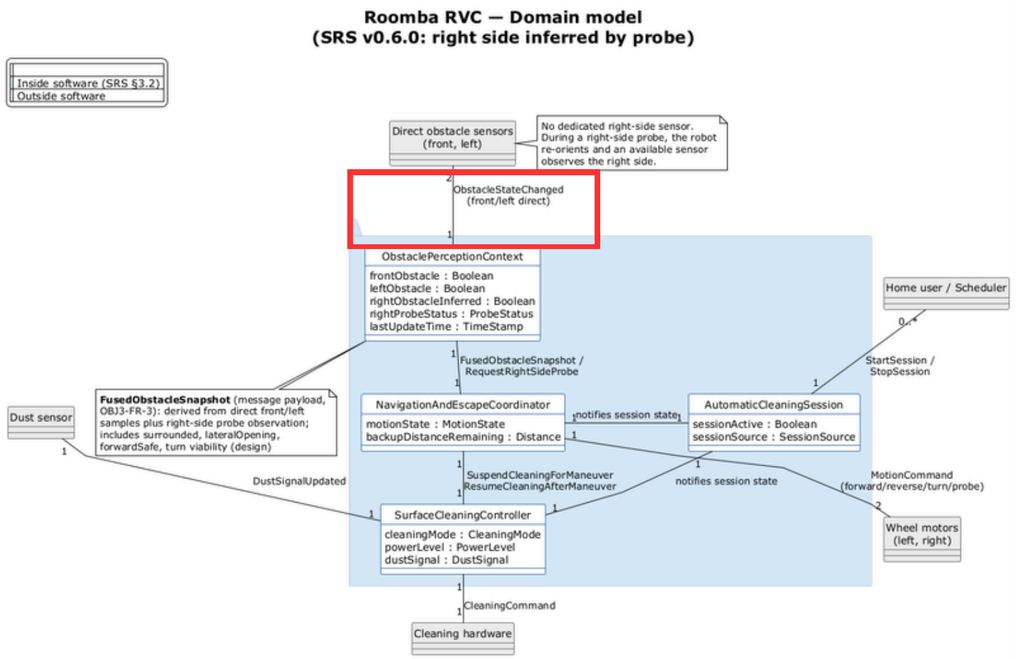


ex) SSD- 5번

VIBE-Domain Diagram

Domain Diagram에는 5가지의 변경사항, 4가지 추가사항, 3가지의 삭제 사항 발생

요약 : 우측 센서 관련 제거, 우측 확인 관련 속성 및 기능 추가, 센서와의 multiplicity 변경



추가 사항

- `rightObstacleInferred` 속성이 추가되었습니다.
- 탐색 자세(probe-pose) 관측치의 상태를 나타내는 `rightProbeStatus` 속성이 추가되었습니다.
- `ObstaclePerceptionContext` 와 `NavigationAndEscapeCoordinator` 간의 `RequestRightSideProbe` 통신이 추가되었습니다.
- 전용 우측 센서는 없으며, 로봇의 방향을 재설정하고 가용한 센싱을 활용하여 우측 상태를 추론한다는 점을 설명하는 참고(note) 문구가 추가되었습니다.

삭제 / 제거 사항

- 외부 도메인 참여자(external domain participant)로서의 직접적인 우측 장애물 센서가 제거되었습니다.
- 직접적인 `rightObstacle` 속성이 제거되었습니다.
- 3개의 장애물 구역(sectors)이 모두 직접적으로 센서를 통해 감지된다는 가정이 제거되었습니다.

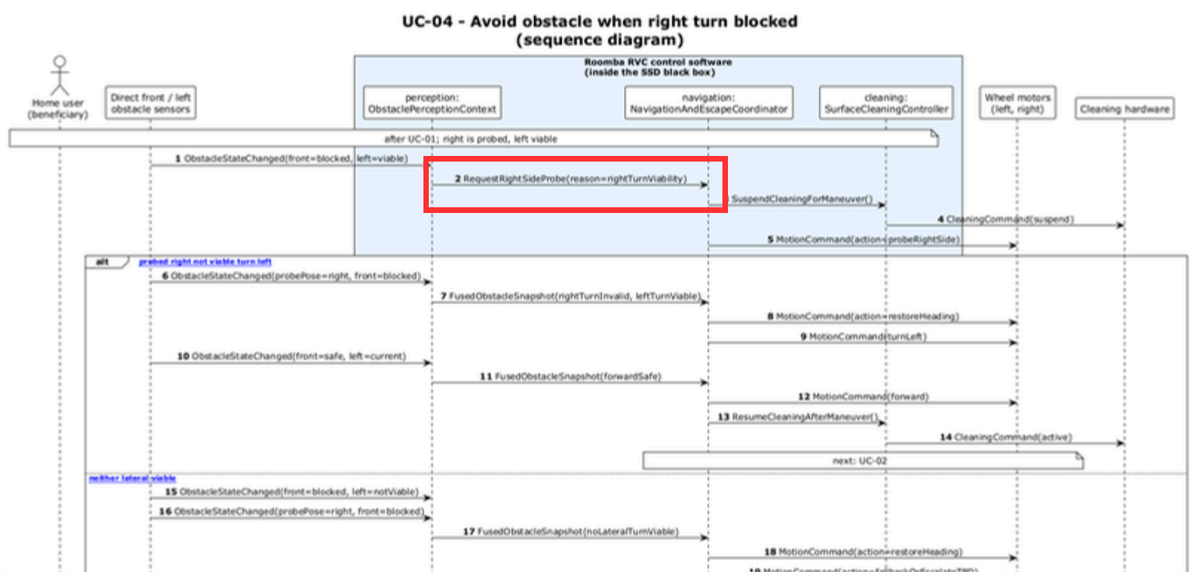
변경 사항

- 외부 장애물 센서 개념이 전면/좌측/우측 센서에서 직접적인 전면/좌측 센서만 사용하는 것으로 변경되었습니다.
- `ObstaclePerceptionContext` 는 이제 직접적인 우측 상태 대신 추론된 우측 상태를 유지합니다.
- 융합된 장애물 정보(fused obstacle picture)는 이제 직접적인 전면/좌측 관측치와 우측 탐색 관측치를 결합합니다.
- 내비게이션과 바퀴의 관계(navigation-to-wheel relationship)에 상위 수준 모션 명령으로서 탐색을 위한 방향 재설정(probe re-orientation)이 명시적으로 포함되었습니다.
- 장애물 센서에서 `ObstaclePerceptionContext` 로의 다중성(Multiplicity)이 `3 – 1` 에서 `2 – 1` 로 변경되었습니다.

VIBE-Sequence Diagram

Sequence Diagram에는 5가지의 변경사항, 4가지 추가사항, 3가지의 삭제 사항 발생

요약 : 우측 센서 관련 제거, 우측 확인 관련 객체, 상호작용 추가



• • •

ex) SD - 4번

변경 사항

- UC-02, UC-03, UC-04, UC-05, UC-07, UC-08 및 UC-09는 이제 일반적인 전면/좌측/우측 장애물 센서 대신 직접적인 **전면 / 좌측** 장애물 센서를 사용합니다.
- UC-03 및 UC-04는 이제 우회전/좌회전 선택 전에 `ObstaclePerceptionContext`가 우측 탐색(right-side probe)을 요청하는 과정을 보여줍니다.
- UC-05는 이제 포위(surrounded) 감지를 '직접적인 전면/좌측 차단 + 탐색된 우측 차단'으로 보여주며, 이후 필수적인 후진 구간(reverse segment)이 이어집니다.
- UC-08은 이제 유효하지 않거나(invalid) 오래된(stale) 직접적인 센서 데이터 외에도, 유효하지 않거나 오래된 탐색 자세 관측치(probe-pose observations)를 처리합니다.
- UC-09는 이제 `FusedObstacleSnapshot(...)` 이전에 직접적인 전면/좌측 샘플과 우측 탐색 자세 관측치가 결합되는 과정을 보여줍니다.

추가 사항

- `RequestRightSideProbe(...)` 객체 상호 작용이 추가되었습니다.
- `MotionCommand(action=probeRightSide)`, `MotionCommand(action=restoreHeading)` 및 `MotionCommand(action=restoreEscapeHeading)` 상호 작용이 추가되었습니다.
- `ObstacleStateChanged(probePose=right, front=...)`를 사용하는 탐색 자세 관측치가 추가되었습니다.
- 포위 탈출(surrounded escape)에는 후진이 필요하며, 180도 회전 후 전진하는 방식으로 대체되지 않는다는 참고(note) 문구가 추가되었습니다.

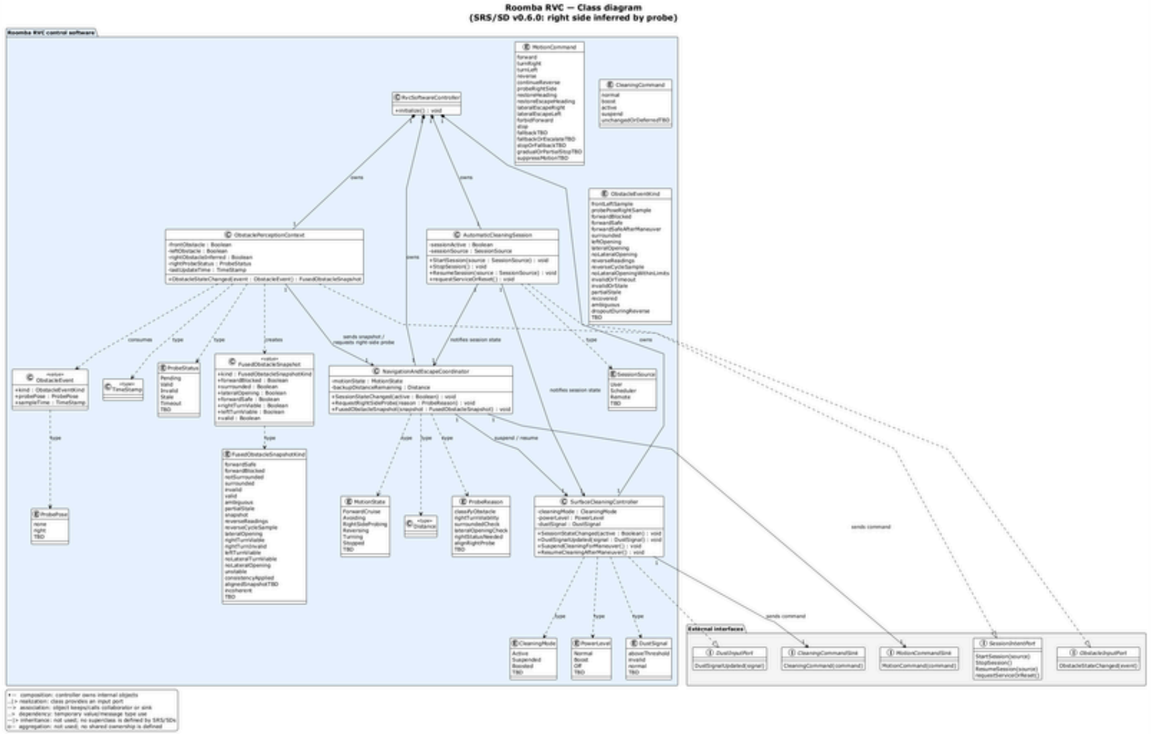
삭제 / 제거 사항

- 직접적인 우측 장애물 센서에 대한 가정이 제거되었습니다.
- 직접적인 `rightBlocked`, `rightInvalid`, `rightTurnViable` 센서 업데이트 관련 문구가 제거되었습니다.
- 다이어그램이 전면/좌측 직접 센싱에 구체적으로 의존하는 부분에서 일반적인 `Obstacle sensors` 참여자(participant) 이름이 제거되었습니다.

VIBE-Class Diagram

Class Diagram에는 5가지의 변경사항, 6가지 추가사항, 4가지의 삭제 사항 발생

요약 : 우측 센서 관련 제거, 우측 확인 관련 상호작용 및 속성 추가



변경 사항

- `ObstaclePerceptionContext` 는 이제 직접적인 `rightObstacle` 대신 `rightObstacleInferred` 및 `rightProbeStatus` 를 저장합니다.
- `NavigationAndEscapeCoordinator` 는 이제 `RequestRightSideProbe(reason : ProbeReason)` 를 수신합니다.
- `MotionCommand` 에는 이제 탐색을 위한 방향 재설정(probe re-orientation) 명령인 `probeRightSide` , `restoreHeading` , `restoreEscapeHeading` 이 포함됩니다.
- `ObstacleInputPort` 는 이제 전면/좌측/우측 센서 업데이트 대신 직접적인 전면/좌측 업데이트 및 탐색 자세 관측치(probe-pose observations)를 나타냅니다.
- `ObstacleEventKind` 값은 이제 직접적인 우측 센서 이벤트 대신 전면/좌측 샘플과 우측 탐색 자세 샘플을 사용합니다.

추가 사항

- 대기 중(pending)/유효함(valid)/유효하지 않음(invalid)/오래됨(stale)/시간 초과(timeout) 등 탐색 상태를 나타내는 `ProbeStatus` 열거형(enum)이 추가되었습니다.
- 우측 탐색이 요청된 이유를 나타내는 `ProbeReason` 열거형이 추가되었습니다.
- 우측으로 방향을 튼 상태에서 측정한 관측치를 표시하기 위한 `ProbePose` 열거형이 추가되었습니다.
- `ObstacleEvent` 에 `probePose : ProbePose` 필드가 추가되었습니다.
- 내비게이션/인식(perception)/이벤트 값 객체(value objects)에서 탐색 관련 열거형으로 연결되는 의존성 링크(dependency links)가 추가되었습니다.
- UC-04의 탐색 시간 초과(probe-timeout) 시퀀스 다이어그램 경로와 일치하도록 `MotionCommand` 에 `stopOrFallbackTBD` 가 추가되었습니다.

삭제 / 제거 사항

- 직접적인 `rightObstacle` 속성이 제거되었습니다.
- `rightBlocked` 및 `rightInvalidForOpening` 과 같은 직접적인 우측 센서 이벤트 이름이 제거되었습니다.
- 직접적인 다방향 센싱(multi-side sensing)을 암시하는 `rawUpdates` , `frontUpdate` , `sideUpdate` 및 `sensorSnapshot` 과 같은 기존의 일반적/업데이트 이벤트 이름이 제거되었습니다.
- `probeOrBackupTBD` 명령이 더 명확한 탐색(probe) 및 대체 동작(fallback) 명령 이름으로 교체되면서 제거되었습니다.

VIBE-Code

Code에는 5가지의 변경사항, 6가지 추가사항, 4가지의 삭제 사항 발생

요약 : 우측 센서 관련 제거, 우측 확인 관련 추가

변경 사항

- 직접적인 우측 장애물 상태가 추론된(inferred) 우측 상태로 변경되었습니다.
- 전방 장애물 회피 시 이제 우측 또는 좌측을 선택하기 전에 우측을 탐색(probe)합니다.
- 포위(Surrounded) 감지는 이제 `frontObstacle && leftObstacle && rightObstacleInferred`를 사용합니다.
- 탐색 자세 관측치(Probe-pose observations)는 `ProbePose::right`와 해당 자세에서의 전면 센서 결과를 사용하여 `ObstacleStateChanged(...)`로 표현됩니다.
- 시뮬레이터와 테스트에 이제 탐색 기반의 우측 확인 및 포위 시 후진 탈출(surrounded reverse escape) 과정이 포함됩니다.

추가 사항

- `Types.hpp`에 `ProbeStatus`, `ProbeReason` 및 `ProbePose`가 추가되었습니다.
- `ObstaclePerceptionContext`에 `rightObstacleInferred_` 및 `rightProbeStatus_`가 추가되었습니다.
- `NavigationAndEscapeCoordinator`에 `RequestRightSideProbe(...)`가 추가되었습니다.
- `probeRightSide`, `restoreHeading`, `restoreEscapeHeading`, `fallbackOrEscalateTBD` 및 `stopOrFallbackTBD`를 위한 모션 명령(Motion commands)이 추가되었습니다.
- 전면 + 좌측 + 탐색된 우측이 모두 차단되었을 때 `surrounded`(포위됨) 상태로 전환되는 것에 대한 단위 테스트 커버리지가 추가되었습니다.
- 우측 탐색 개방(clear), 우측 탐색 차단(blocked), 탐색 시간 초과(timeout), 회전 전 후진 탈출(reverse-before-turn escape)에 대한 시뮬레이터 시나리오가 추가되었습니다.

삭제 / 제거 사항

- 직접적인 `rightObstacle_` 저장이 제거되었습니다.
- `rightBlocked` 및 `rightInvalidForOpening`과 같은 직접적인 우측 센서 이벤트 이름이 제거되었습니다.
- `rawUpdates`, `frontUpdate`, `sideUpdate`, `sensorSnapshot` 및 `forwardBlockedNotSurrounded`와 같은 기존의 일반적인 장애물 이벤트 이름이 제거되었습니다.
- `probeOrBackupTBD`가 더 명확한 탐색(probe) 및 대체 동작(fallback) 명령으로 교체되면서 제거되었습니다.

VIBE-Unit Test

Unit Test에는 5가지의 변경사항,
6가지 추가사항, 4가지의 삭제 사항 발생

요약 : 우측 센서 관련 테스트 제거, 우측 확인 과정 관련 테스트 추가

```
Running main() from C:\Users\cheky\Cursor\cpp\build\_deps\goog
[==========] Running 41 tests from 5 test suites.
[----------] Global test environment set-up.
[----------] 4 tests from AutomaticCleaningSessionTest
[ RUN      ] AutomaticCleaningSessionTest.StartSessionEnablesN
[       OK ] AutomaticCleaningSessionTest.StartSessionEnablesN
[ RUN      ] AutomaticCleaningSessionTest.StopSessionStopsMoti
[       OK ] AutomaticCleaningSessionTest.StopSessionStopsMoti
[ RUN      ] AutomaticCleaningSessionTest.ResumeSessionEnables
[       OK ] AutomaticCleaningSessionTest.ResumeSessionEnables
[ RUN      ] AutomaticCleaningSessionTest.RequestServiceOrRese
[       OK ] AutomaticCleaningSessionTest.RequestServiceOrRese
[----------] 4 tests from AutomaticCleaningSessionTest (1 ms t
```

• • •

```
[ RUN      ] SurfaceCleaningControllerTest.SuspendCleaningForMan
[       OK ] SurfaceCleaningControllerTest.SuspendCleaningForMan
[ RUN      ] SurfaceCleaningControllerTest.ResumeCleaningAfterMar
[       OK ] SurfaceCleaningControllerTest.ResumeCleaningAfterMar
[ RUN      ] SurfaceCleaningControllerTest.ResumeCleaningAfterMar
[       OK ] SurfaceCleaningControllerTest.ResumeCleaningAfterMar
[----------] 8 tests from SurfaceCleaningControllerTest (1 ms to

[----------] Global test environment tear-down
[==========] 41 tests from 5 test suites ran. (14 ms total)
[  PASSED  ] 41 tests.
```

GCC Code Coverage Report

Directory: ./
Coverage: low: ≥ 0% medium: ≥ 75.0% high: ≥ 90.0%

Coverage	Exec / Excl / Total
Lines: 87.7%	293 / 0 / 334
Functions: 97.4%	37 / 0 / 38
Branches: 73.8%	110 / 0 / 149

List of functions
File
Lines
Functions
Branches
include/rvc/Ports.hpp
100.0% 5 / 0 / 5
100.0% 5 / 0 / 5
-% 0 / 0 / 0
src/AutomaticCleaningSession.cpp
100.0% 24 / 0 / 24
100.0% 6 / 0 / 6
-% 0 / 0 / 0
src/NavigationAndEscapeCoordinator.cpp
96.5% 111 / 0 / 115
100.0% 9 / 0 / 9
81.5% 75 / 0 / 92
src/ObstaclePerceptionContext.cpp
76.7% 89 / 0 / 116
100.0% 3 / 0 / 3
60.0% 24 / 0 / 40
src/RvcSoftwareController.cpp
90.9% 20 / 0 / 22
88.9% 8 / 0 / 9
50.0% 3 / 0 / 6
src/SurfaceCleaningController.cpp
84.6% 44 / 0 / 52
100.0% 6 / 0 / 6
72.7% 8 / 0 / 11

Created using GCOVR (Version 8.6) at 2026-06-02 08:33:30

변경 사항

- 단위 테스트는 이제 직접적인 우측 센서 이벤트 대신 업데이트된 탐색 기반(probe-based) 장애물 모델을 사용합니다.
- 전방 장애물 테스트는 이제 우측 상태가 최신이 아니거나(not fresh) 유효하지 않을 때 `probeRightSide` 명령을 기대합니다(expect).
- 회피 테스트는 이제 탐색 후 최종 우회전/좌회전 명령을 내리기 전에 `restoreHeading` 을 기대합니다.
- 후진 탈출 테스트는 이제 동작 흐름에 탐색/탈출 방향 복원(probe/escape heading restoration)이 포함될 경우, 후진을 계속하기 전에 `restoreEscapeHeading` 을 기대합니다.
- 시뮬레이터 기반의 예상 결과(expectations)가 업데이트되어, 포위 탈출(surrounded escape) 시 측면 탈출(lateral escape) 전에 여전히 후진이 필요하도록 변경되었습니다.

추가 사항

- `ObstacleEvent` 를 통한 `ProbePose::right` 관측치에 대한 테스트가 추가되었습니다.
- 우측 탐색 결과가 개방(clear) 상태일 때 우회전 가능(right-turn viability)으로 이어지는지에 대한 테스트가 추가되었습니다.
- 우측 탐색 결과가 차단(blocked) 상태일 때 좌측이 열려 있으면 좌회전 가능으로 이어지는지에 대한 테스트가 추가되었습니다.
- 전면 + 좌측 + 탐색된 우측이 모두 차단되었을 때 `surrounded` (포위됨) 상태로 전환되는지에 대한 테스트가 추가되었습니다.
- 회피 탐색(avoidance probing)을 수행하기 전에 `RequestRightSideProbe(...)` 가 청소를 일시 중단(suspending cleaning)하는지에 대한 테스트가 추가되었습니다.
- C++ 커버리지 스크립트를 사용한 커버리지 테스트 실행(coverage run)이 추가되었습니다.

삭제 / 제거 사항

- `rightBlocked` 및 `rightInvalidForOpening` 과 같은 직접적인 우측 센서 이벤트에 대한 테스트 참조가 제거되었습니다.
- `frame` , `rawUpdates` , `frontUpdate` , `sideUpdate` , `sensorSnapshot` 및 `forwardBlockedNotSurrounded` 와 같은 기존의 일반적인 이벤트 이름에 대한 테스트 참조가 제거되었습니다.
- `probeOrBackupTBD` 에 대한 기존 예상 결과(expectation)가 제거되었습니다. 테스트는 이제 명시적인 탐색(probe), 복원(restore), 대체 동작(fallback) 또는 정지/대체 동작(stop/fallback) 명령을 사용합니다.

VIBE-Simulator

Simulator에는 5가지의 변경사항, 7가지 추가사항, 4가지의 삭제 사항 발생

요약 : 우측 센서 관련 테스트 제거, 우측 확인 과정 관련 테스트 추가

```
RVC Software Controller Simulator
Running 30 scripted scenarios
Use --verbose to print actions and command traces for every scenario.

[PASS] TC-01  Initialize enters safe inactive state
[PASS] TC-02  Start and cruise forward while cleaning
[PASS] TC-03  Dust boost returns to normal while active
[PASS] TC-04  Dust boost is deferred while session is inactive
[PASS] TC-05  Invalid dust signal keeps normal cleaning
[PASS] TC-06  Stop session stops motion and suspends cleaning
[PASS] TC-07  Resume after stop re-enters forward cleaning
[PASS] TC-08  Service/reset request enters inactive safe state
[PASS] TC-09  Invalid obstacle data triggers full fault stop
[PASS] TC-10  Partial stale obstacle data uses gradual stop
[PASS] TC-11  Recovery snapshot is accepted without new command
```

• • •

```
[PASS] TC-23  Ambiguous obstacle snapshot waits for later policy
[PASS] TC-24  Front left sample builds consistency snapshot only
[PASS] TC-25  Ambiguous probe alignment waits for later policy
[PASS] TC-26  Forward safe after maneuver resumes cleaning
[PASS] TC-27  Forward blocked after maneuver requests right-side probe
[PASS] TC-28  Going back then turning: reverse before right escape
[PASS] TC-29  Going back then left turning after left opening
[PASS] TC-30  Full mission path with boost, trap escape, resume, stop

Summary: 30 / 30 scenarios passed.
```

변경 사항

- 시뮬레이터 시나리오는 이제 직접적인 우측 센서 이벤트 대신 우측 탐색 모델(right-side probe model)을 사용합니다.
- 전방 차단(Forward-blocked) 시나리오는 이제 우측 상태 정보가 필요할 때 최종 회전 결정 전에 `probeRightSide`를 기대합니다(expect).
- 회피 시나리오는 이제 `turnRight` 또는 `turnLeft` 전에 `restoreHeading`을 기대합니다.
- 포위 탈출(Surrounded escape) 시나리오는 이제 측면 탈출(lateral escape) 전에 후진 이동을 검증합니다.
- 시나리오 설명이 업데이트되어 기존의 직접적인 우측 센서 관련 문구가 제거되었습니다.

추가 사항

- 시뮬레이터 장애물 동작(obstacle actions)에 `ProbePose::right` 데이터가 추가되었습니다.
- 복사된 시뮬레이터 소스 산출물(artifact)인 `rvc_simulator.cpp`가 추가되었습니다.
- 우측 탐색 결과가 개방(clear) 상태일 때 우회전으로 이어지는 시나리오가 추가되었습니다.
- 우측 탐색 결과가 차단(blocked) 상태일 때 좌회전으로 이어지는 시나리오가 추가되었습니다.
- `stopOrFallbackTBD`를 사용하는 탐색 시간 초과(timeout) 시나리오가 추가되었습니다.
- 전면 + 좌측 + 탐색된 우측이 모두 차단되었을 때 포위된(surrounded) 상태로 전환되어 후진을 시작하는 시나리오가 추가되었습니다.
- 해당 폴더에 실행 스크립트인 `run-simulator.ps1` 및 `run-simulator.bat`가 추가되었습니다.

삭제 / 제거 사항

- `rightBlocked` 및 `rightInvalidForOpening`과 같은 직접적인 우측 센서 시뮬레이터 이벤트가 제거되었습니다.
- `frame`, `rawUpdates`, `frontUpdate`, `sideUpdate`, `sensorSnapshot` 및 `forwardBlockedNotSurrounded`와 같은 기존의 일반적인 이벤트 이름이 제거되었습니다.
- `probeOrBackupTBD`에 대한 기존 시뮬레이터 예상 결과(expectation)가 제거되었습니다.
- 물리적인 우측 장애물 센서를 암시하는 시나리오 문구가 제거되었습니다.

Traditional 진행

변경된 요구 사항: 우측 장애물 센서 삭제

분석 단계에서 우측 센서 데이터를 직접 이용해 감지/회피하는 흐름 제거

설계 단계에서 센서 구성과 객체 책임을 수정해 우측 센서 의존성 제거

구현 단계에서 우측 센서 없이 판단하도록 로직과 테스트 데이터 정리

마지막으로 Unit Test, Simulator 로 기존 주행 동작이 유지되는지 확인

VIBE 진행

경험 및 느낀점 : VIBE

harness 기능이 제공되면 반드시 사용(rules, skills, commands)
> 체감 토큰 사용량(950K → 422K 약 50%), 체감 소요시간(10H → 3H)

한번에 해달라가 아닌, 작은 절차(srs, usecase ...)를 밟아가며 사용

skill에서 cnc하게 하라는 내용 필수

혹시 실수 할 수 있으니, OOAD 등 개념을 반드시 상세하게 설명.

변경 내용을 기록& 쉽게 볼 수 있는 양식이 필요함.
(appendix, human readable, 노션 등 mcp 연결 등등)

중간 검사 시, 쉽게 볼 수 있는 규칙/양식 필요함.
(prompt 단위에서 변경 사항 요약, md file 상 hyperlink 등등)

코드가 쓸데없이 커지지 않도록 조절하는 방식 필요함