보 고 서

무인주차관제 시스템 및 시뮬레이션 개발

윈도우프로그래밍 기말 프로젝트

팀명 :	혼코
학과 :	컴퓨터공학부
학번 :	201711026
이름 :	강성원

목 차

I. 프로젝트 개요

1. 프로젝트 기획 배경

오늘날 무인주차관제 시스템은 관리자가 요구한 기능을 분석 후 추가 구현하여 시스템이 점진적으로 발전이 되고 있다. 하지만 시스템 개발이 활성화됨에 따라 소프트웨어 테스트 과정 없이 적용 시 예상하지 못한 시스템 및 데이터 오류가 발생하게 되면서 해당 주차장의 운영이 중단되어 경제적, 시간적 손해와 주변 구역에 불법주차 및 주차난을 초래하게 된다. 경제적, 시간적 소모와 문제점을 초래될 수 있는 요인들을 해결하기 위해서는 개발된 무인주차관제 시스템에 대한 신뢰성과 성능을 검증할 수 있는 시뮬레이션이 필요하다.

본 프로젝트는 무인 주차관제 시스템 및 시뮬레이터를 구현하여 개발된 무인 주차관제 시스템에 대한 무결성과 성능, 신뢰성을 검증하기 위해 시뮬레이터를 구현한다. 이로 통해서 무인 주차관제 시스템은 구현된 시뮬레이터로 통해 생성된 차량 데이터량만큼 이벤트 처리하면서 데이터에 대한 무결성과 시스템 성능, 신뢰성을 검증하는 프로그램을 만들고자 한다.

2. 프로젝트 목적과 요구사항

최종 구현한 프로그램은 무인주차관제 시스템과 시뮬레이터를 구현하여 시뮬레이션에 따라 무인주차관제 시스템이 정상적으로 동작해야 한다. 따라서 본 프로젝트 프로그램은 아래와 같은 사항을 요구한다.

(1) 기능적 요구사항

- 사용자가 입력한 차량의 대수만큼 시뮬레이터에서 차량 데이터를 생성한다.
- 사용자가 입력한 이벤트 최소/최대 발생시간 범위에서 이벤트를 발생한다.
- 이벤트로 발생된 입차/출차한 차량 및 시스템 데이터를 실시간으로 저장한다.
- 이벤트로 발생된 데이터를 실시간으로 처리할 수 있도록 자동화한다.
- 시스템 오류 또는 결함이 발생 시 동작 중인 시스템과 시뮬레이션을 중지시키고, 오류 메시지를 출력한 후 재동작할 수 있도록 전체적으로 초기화한다.

(2) 비기능적 요구사항

- 시뮬레이션 이벤트 최소 발생시간은 100ms부터 동작한다.
- 시뮬레이션에 생성할 차량의 대수는 사용자의 입력에 따라서 차량 데이터들을 생성한다.
- 로그 데이터 파일은 60,000 라인까지 데이터들을 저장한다.
- 무인 주차관제 시스템은 최소 100분 이상 시스템 오류 및 결함없이 실행한다.

2. 프로젝트 일정

1. 프로젝트 개발일정

무인주차관제 시스템 및 시뮬레이션 개발을 진행하면서 개발 일정을 아래와 같이 Table 1에 나타낸다.

일시	내용	총 담당
4월 1주차	• 프로그램 아이디어 선정 및 관련 자료 조사 프로그램 구조 설계	**강성원**
4월 3주차	• 시뮬레이션 기능을 추가하기로 결정 시뮬레이션 기능을 추가로 인한 전체 재설계	
4월 3주차	• 시뮬레이션 기능을 추가하기로 결정 • 시뮬레이션 기능을 추가로 인한 전체 재설계	
4월 4주차	• 프로그램 디자인 설계 • 제안서 작성 및 제출	
4월 2주차	• 주차공간화면 디자인 • 설정화면 디자인 • 주차이력화면 디자인	

일시	내용	총 담당
5월 3주차	• 설정화면 기능 구현	**강성원**
5월 5주차	• 싱글톤&시뮬레이션 차량 데이터 생성 구현	
5월 5주차	• 시뮬레이션 입차 이벤트 구현 • 주차공간화면 1,2 층 전환 시 UI 오류 발견	
6월 1주차	• 시뮬레이션 출차 이벤트 구현	
6월 2주차	• 중간 프로그램 테스트 • 발견된 오류 리스트 작성	
6월 3주차	• 발견된 오류 예외처리 및 코드 수정	
6월 4주차	• 로그 및 차량 데이터 저장 기능 구현 • 차량 이력화면 기능 구현 • 프로그램 최종 테스트 및 점검	

Table 1. 개발 일정

III. 프로젝트 내용

1. 프로젝트 구조

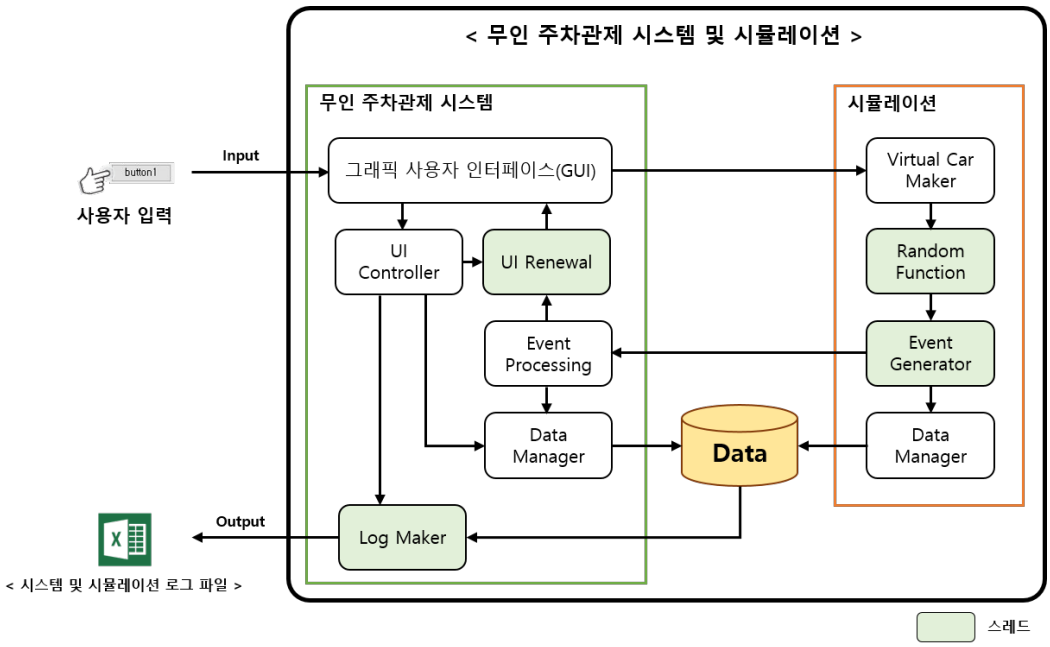

Figure 1. 무인 주차관제 시스템 및 시뮬레이션 구조

그림 1는 (1) 그래픽 사용자 인터페이스(GUI)를 백엔드 기능을 구성하여 무인 주차관제 시스템과 시뮬레이션 기능을 나뉘어 구조를 나타내었다. 무인 주차관제 시스템의 내부 구조는 사용자의 GUI 입력에 따라서 UI 기능을 실행하여 UI 갱신 및 출력과 로그 생성, 데이터 저장으로 구성된다. 그리고 GUI에서 시뮬레이션의 기능을 활성화 시 시작 또는 중지를 수행하여 가상 차량 데이터를 생성하고, 랜덤 값을 출력하여 값에 따라서 이벤트 발생 시간 출력 및 가상 차량 데이터를 전송한다. 무인 주차관제 시스템과 시뮬레이션에서 발생된 데이터는 데이터 관리 클래스로 통해서 저장하고, 로그 생성 GUI 활성화 시 저장된 데이터들을 문서 파일로 생성하여 저장한다.

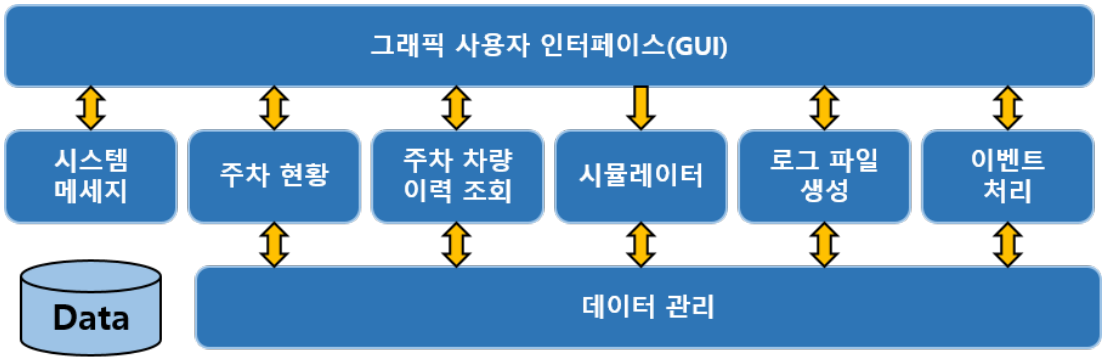

Figure 2. 소프트웨어 상세 구성도

그림 2은 (1) 그래픽 사용자 인터페이스(GUI)와 (2) 소프트웨어(SW) 구조기반으로 설계된 무인 주차관제 시스템 및 시뮬레이션 상세 구성도이다. 시스템 구성은 계층적으로 구성되어 기능 변경 및 추가가 용이하도록 설계되었다. 최상층은 사용자의 입력을 받는 C# 기반의 WPF UI로 구성된다. 중간층은 입력을 받은 시스템이 실제 수행되어야 할 기능으로 시스템 메세지, 주차 현황, 주차 차량 이력 조회, 시뮬레이터, 로그 파일 생성, 이벤트 처리로 구성된다. 최하층은 시스템 동작 시 수집되는 데이터들을 저장하는 데이터 저장과 시스템 또는 시뮬레이션에서 발생된 데이터들을 관리할 수 있게 구성되어 있다.

2. 구현

제안한 프로젝트에 대해서 구현 가능성과 타당성을 검토 및 분석하여 아래와 같이 작성하였다. 사용자에게 제공할 UI 프레임워크와 데이터 및 UI 처리 등 동시 작업을 수행할 수 있는 병행수행 기법, 데이터 Save 또는 Load 할 수 있는 기법, 로그 데이터 파일 저장, 가상 데이터 생성에 대해서 제시한다.

(1) WPF

- GUI 프로그램 개발은 윈도우에서 작동하는 GUI 프로그램을 개발하기 위한 프로그래밍 모델인 WPF UI 프레임워크를 사용한다.
- 각 UI 요소에 대한 컨트롤 구성과 디자인 삽입, 데이터 바인딩을 사용하여 무인 주차관제 시스템의 레이아웃(프론트엔드)을 배치한다.
- 배치된 UI 요소마다 기능(백엔드)에 맞게 구현하여 사용자가 입력된 조작에 따라서 수행한다.

(2) Thread

- 무인 주차관제 시스템에서 시뮬레이션을 동작하면 프로그램의 성능이 전체적으로 저하되므로 동일 시간에 많은 작업을 수행하기 위해 작업에 따라서 각각 쓰레드를 사용한다.
- 각 쓰레드의 수행은 UI 갱신 및 출력, 시뮬레이션 동작, 데이터 저장, 예외처리를 병행한다.

(3) Singleton Pattern

- 무인 주차관제 시스템에서 발생된 데이터를 저장하고, 다른 클래스 및 폼에서 저장된 데이터를 사용하기 위해 싱글톤 패턴(Singleton Pattern) 기법을 사용한다.
- 싱글톤 패턴이란 프로그램이 시작될 때 클래스가 단 한번만 메모리를 할당하고, 전역 인스턴스를 만들어 사용하는 패턴을 말한다. 싱글톤 패턴은 메모리 낭비를 방지하고, 다른 클래스와 데이터를 공유하기가 쉽다.

(4) 로그 저장

- 무인 주차관제 시스템 및 시뮬레이션에서 발생 또는 생성된 데이터를 마이크로소프트사의 마이크로오피스 엑셀 문서 파일을 생성하여 모두 저장한다.
- 데이터는 시뮬레이션에서 생성된 차량 정보와 이벤트 발생 로그를 저장하고, 무인 주차관제 시스템에서 발생된 입차/출차에 대한 차량 정보와 주차요금, 상태메세지, 주차이력들을 저장한다.

(5) 가상 차량 데이터

- 무인 주차관제 시스템을 시뮬레이션하기 위해 입차/출차에 대한 차량 데이터로 시스템을 동작시켜야하므로 가상 차량 데이터를 생성한다.
- 사용자가 입력한 차량 대수만큼 가상 차량 데이터를 생성하며, 차량 데이터는 차량 번호, 입차/출차 날짜시간, 지불한 총 주차요금의 정보로 구조체형태로 지닌다.
- 차량 번호는 현 자동차관리법에 맞게 차량의 차종, 용도, 차량등록번호의 기준으로 구성하여 데이터를 난수로 생성한다.

3. 클래스 설계

소프트웨어의 개발에 있어 필요한 클래스에 대한 설계이며, 아래와 같이 클래스를 포함하고 있는 레이아웃 단위로 서술하였다.

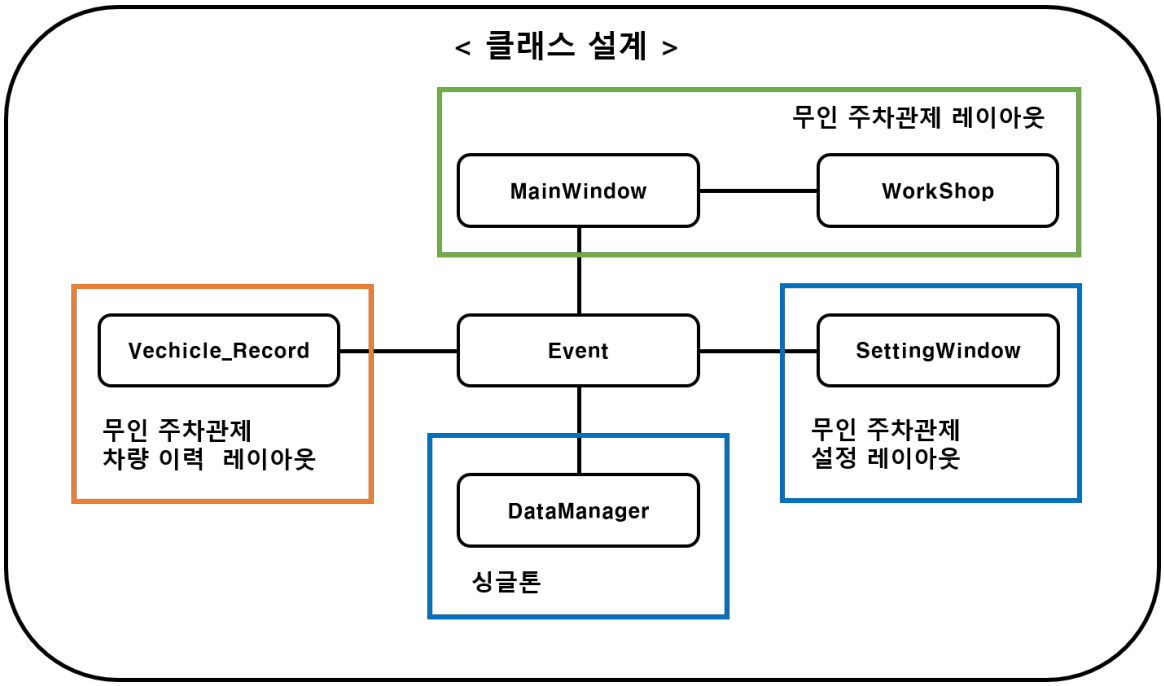

Table 3. 클래스 설계

4. 무인 주차관제 시스템 레이아웃

(1) 무인 주차관제 시스템 레이아웃 기능

- **주차공간** : 차량이 주차된 구역은 빨간불을 표시하고, 비어있는 주차구역은 초록불로 표시
- **상태 메시지** : 주차 현황 기능과 시스템 메세지에 대한 시간, 입차/출차하는 차량 번호, 요금 출력
- **주차 현황** : 현재 주차된 모든 차량에 대한 차량 번호와 입차 시간을 상태 메시지 UI에 출력
- **상태 메세지 삭제** : 상태 메시지 UI에 출력된 데이터 삭제
- **1층/2층** : 각 층마다 주차 공간 화면 UI 제공
- **주차 이력** : 사용자가 입력한 차량 번호로 통해 주차 이력 조회 제공
- **차량 검색** : 차량 리스트 검색 기능을 제공
- **로그 저장** : 입차/출차한 차량 및 시스템 데이터에 대한 로그 저장
- **시뮬레이션** : 무인 주차관제 시스템에 대한 시뮬레이션 기능 On/Off
- **현재 날짜/실시간** : 현재 날짜와 시간을 출력

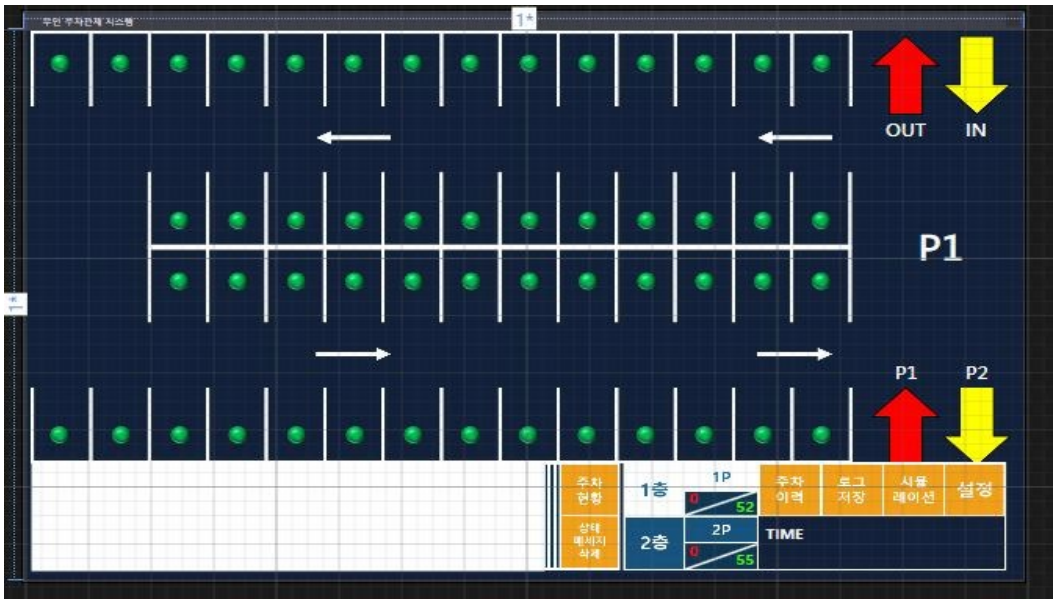

Figure 4. 무인 주차관제 시스템 레이아웃

(2) 무인 주차관제 시스템 레이아웃 내부 소스

[MainWindow.cs]	설명
void myTimer_Tick	▪ 1 초마다 현재 시간과 날짜를 MainWindow 에 배치된 Label 에 저장
ImageBrush Set_Image	▪ 컨테이너의 배경을 이미지로 설정하기 위한 메소드
void Erase_Click	▪ 상태 메시지 Text 초기화
void Trans_Screen_Click	▪ 1/2 층 전환기능
void Show_P_Car_Click	▪ 현재 층에서 주차된 차량들에 대한 차량 번호를 Message 에 출력

Table 2. 무인 주차관제 시스템 레이아웃 내부 소스

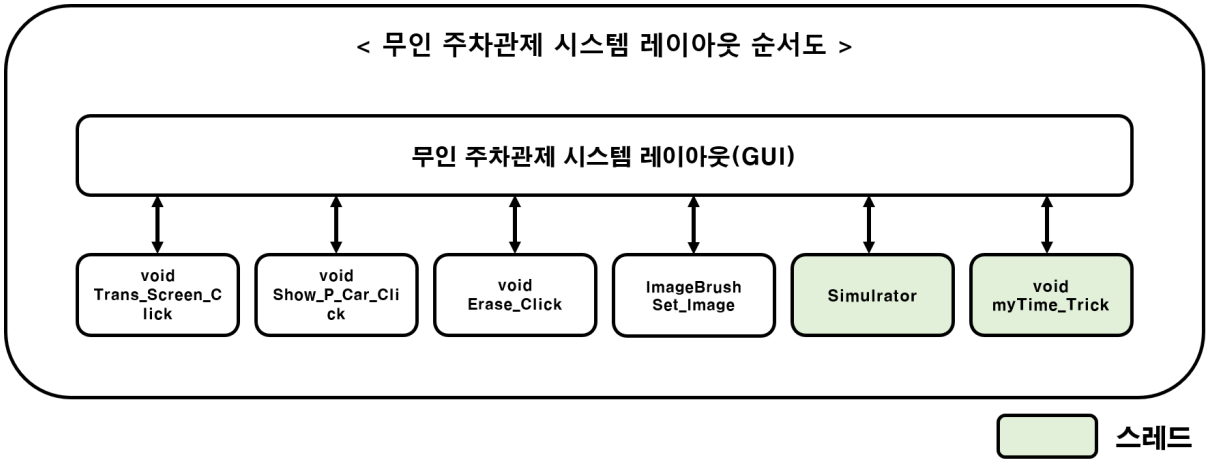

Figure 5. 무인 주차관제 시스템 레이아웃 순서도

(3) 무인 주차관제 시스템 레이아웃 시뮬레이션 내부 주요 소스

[MainWindow.cs]	설명
void Initialize	▪ 싱글톤에 저장된 차량 데이터와 각 층 차량 카운터를 초기화 ▪ 시뮬레이션 버튼 UI 이미지 변경
void generate_simulrator_car_row_data **void get_car_number_center_data_char** **void make_simulator_car_inf**	▪ 설정된 차량 대수만큼 차량번호(00 '가' 0000)를 랜덤 값으로 생성하여 싱글톤에 임시로 저장 ▪ 랜덤 값은 singular_Random_value, plural_Random_value 메소드에서 반환받음 ▪ 마지막으로 차량번호들을 조합한 후 싱글톤에 정의된 구조체 각 Parking_Car 에 차량번호를 저장하고 구초에 선언된 변수들을 초기화

Table 3. 무인 주차관제 시스템 레이아웃 시뮬레이션 내부 소스

(4) 무인 주차관제 시스템 레이아웃 시뮬레이션 내부 주요 소스 2

[MainWindow.cs]	설명
void start_parking_manager_simulator	▪ 시뮬레이션 이벤트가 동작하는 메소드 ▪ 쓰레드로 simulatorEvent 메소드을 호출 ▪ 이벤트 발생시간은 최소~최대 사이의 랜덤 값이며 설정이 가능
void simulatorEvent	▪ 변수 event_mode 에 입차/출차 메소드를 호출 ▪ 차량 데이터는 rnd_index 로 선택되며 event_mode 에 따라 이벤트가 동작 ▪ 시뮬레이션 중지될 때까지 반복해서 쓰레드 동작
void set_park_car	▪ rnd_index 를 car_index 로 전달받아 car_index 에 위치한 구조체에 입차 관련 데이터를 저장 ▪ 입차된 주차 공간 이미지를 빨간불 이미지로 변경 ▪ 입차 시 해당 층의 카운터를 수정
void set_exit_car	▪ 출차 시 int Charge 메소드를 동작 ▪ 출차 시 해당 층의 카운터를 수정 ▪ 출차된 주차 공간 이미지를 초록불 이미지로 변경 ▪ 출차된 차량의 구조체의 데이터를 초기화
int Charge	▪ set_exit_car 메소드에서 호출 ▪ 현재 시간과 출차하려는 차량의 입차시간의 차를 구하여 주차시간을 계산 ▪ 구한 주차시간으로 요금을 측정하고 반환 ▪ 해당 프로그램은 시뮬레이션을 위해 1 초 당 요금을 지불

Table 4. 무인 주차관제 시스템 레이아웃 시뮬레이션 내부 주요 소스 2

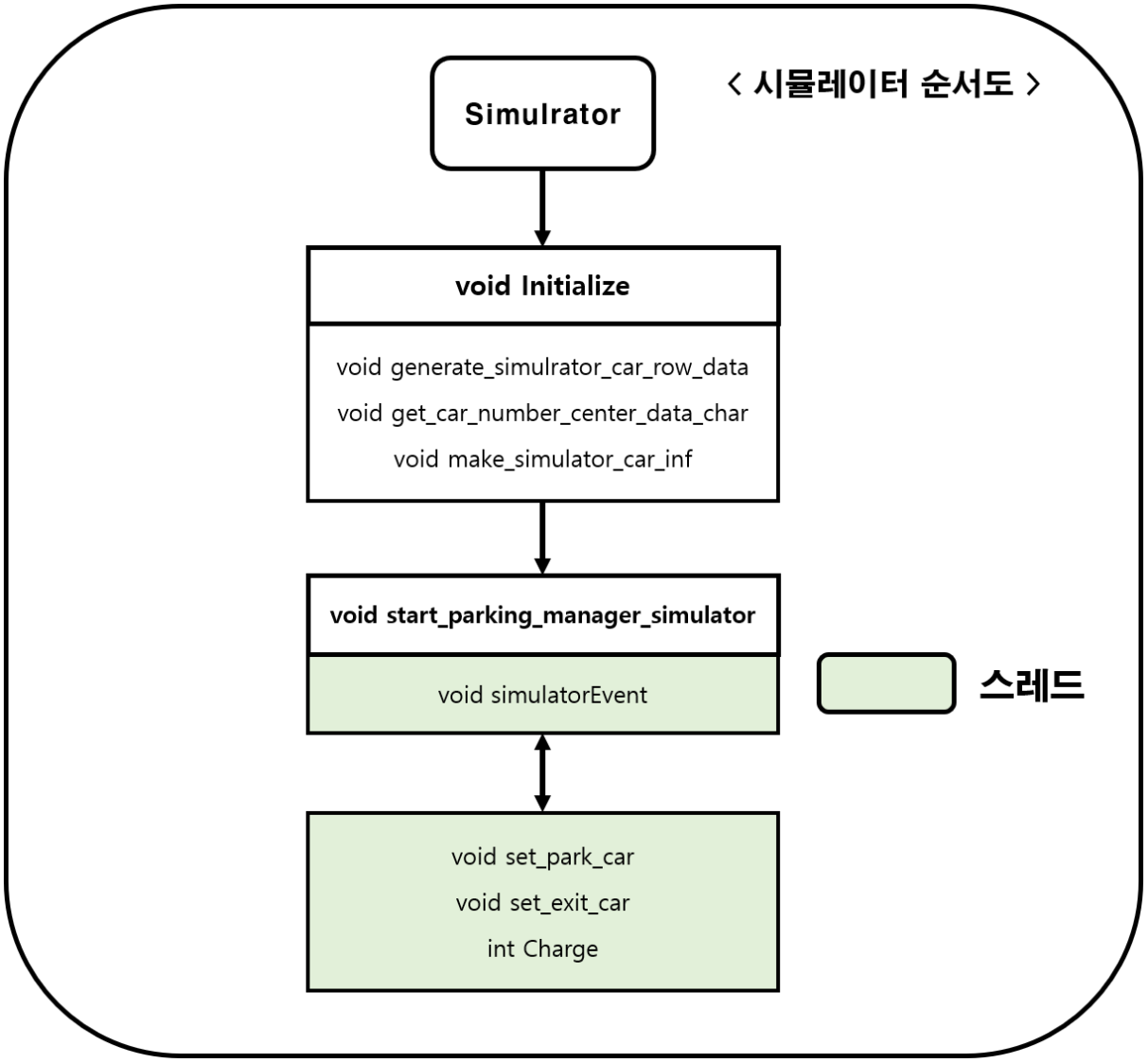

Table 6. 무인 주차관제 시스템 레이아웃 시뮬레이터 순서도

5. 무인 주차관제 시스템 주차 이력 레이아웃

(1) 무인 주차관제 시스템 주차 이력 레이아웃

주차 이력 : 사용자가 입력한 차량 번호로 통해 주차 이력 조회 제공

- **차량 검색** : 차량 리스트 검색 기능을 제공
- **차량 리스트** : 시뮬레이션으로 생성된 데이터 중 입/출차된 차량번호를 리스트로 제공
- **차량 이력** : 차량 리스트에서 선택된 차량에 대한 정보를 제공
- **새로 고침** : 차량 리스트 최신화

Figure 6. 무인 주차관제 시스템 주차 이력 레이아웃

(2) 무인 주차관제 시스템 주차이력 레이아웃 내부 소스

[Vehicle_Record.cs]	설명
void Initialize	▪ void SortList 메소드를 호출 ▪ 리스트 CarNumberlist 의 데이터들을 ComboBox 와 ListBox 에 추가 하는 메소드
void SortList	▪ 리스트 CarNumberList 에 입/출차된 차량들의 차량번호를 추가 ▪ 이 후 CarNumberList 를 정렬
void Vechicle_SelectionChanged	▪ ComBox 와 ListBox 에 선택된 차량번호를 Show_CarInformation 메소드에 전달
void Show_CarInformation	▪ 전달 받은 차량 번호를 선형 탐색으로 조회하여 Label 에 정보를 출력
void Refresh_Click	▪ 새로고침 버튼을 누를 시 동작한다. ▪ Iinitialize 메소드를 호출하고 Label 을 초기화

Table 5. 무인 주차관제 시스템 주차이력 레이아웃 내부 소스

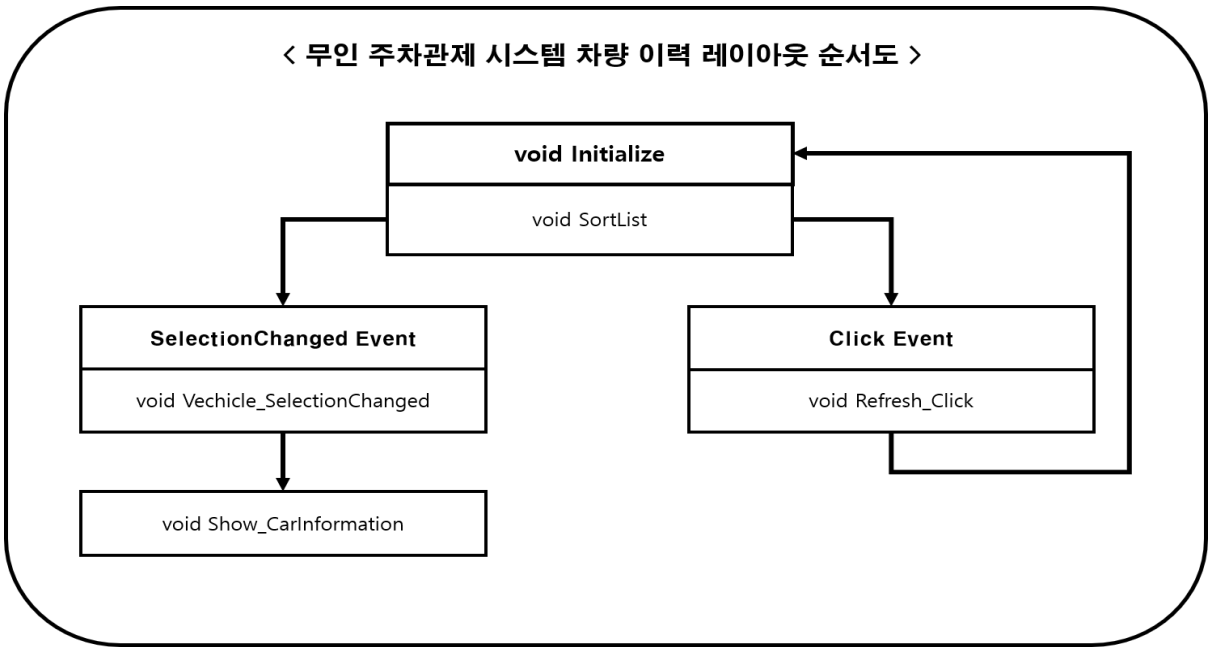

Figure 7. 무인 주차관제 시스템 차량 이력 레이아웃 순서도

6. 무인 주차관제 시스템 설정 레이아웃

(1) 무인 주차관제 시스템 설정 레이아웃

설정 : 주차 요금과 시뮬레이션의 생성할 차량 대수, 이벤트 발생시간 최소/최대 설정

- **생성할 차량 대수** : 시뮬레이션에서 생성할 가상 차량 대수 입력
- **이벤트 발생 시간** : 시뮬레이션의 이벤트 최소/최대 발생 시간 입력
- **30분 당 기본요금** : 무인 주차관제 시스템의 30분 당 기본요금 입력
- **1시간 당 추가되는 기본요금** : 1시간 당 추가되는 기본요금 입력

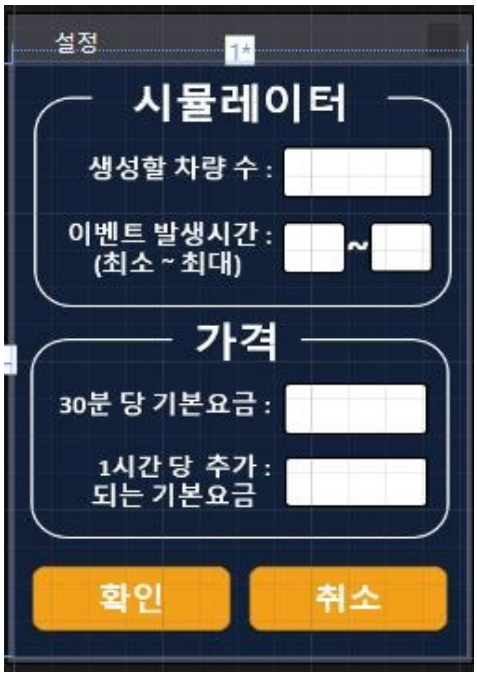

Figure 7. 무인 주차관제 시스템 설정 레이아웃

(2) 무인 주차관제 시스템 설정 레이아웃 내부 소스

[SettingWindow.cs]	설명
void Initialize	▪ 현재 설정된 데이터들을 각 TextBox에 출력
void Setting_Click	▪ 현재 설정화면에 입력된 TextBox들의 내용을 변수에 임시 저장 ▪ 임시 저장된 설정데이터들을 대리자를 통해 GetSettingData 호출
[MainWindow.cs]	**설명**
void Setting_Click	▪ 메인에서 [설정]버튼을 누를 시 설정화면을 실행 ▪ 이때 대리자로 GetSettingData로 지정
void GetSettingData	▪ 설정화면에서 대리자를 통해 전달받은 데이터를 싱글톤 DataManager에 저장

Table 6. 무인 주차관제 시스템 설정 레이아웃 내부 소스

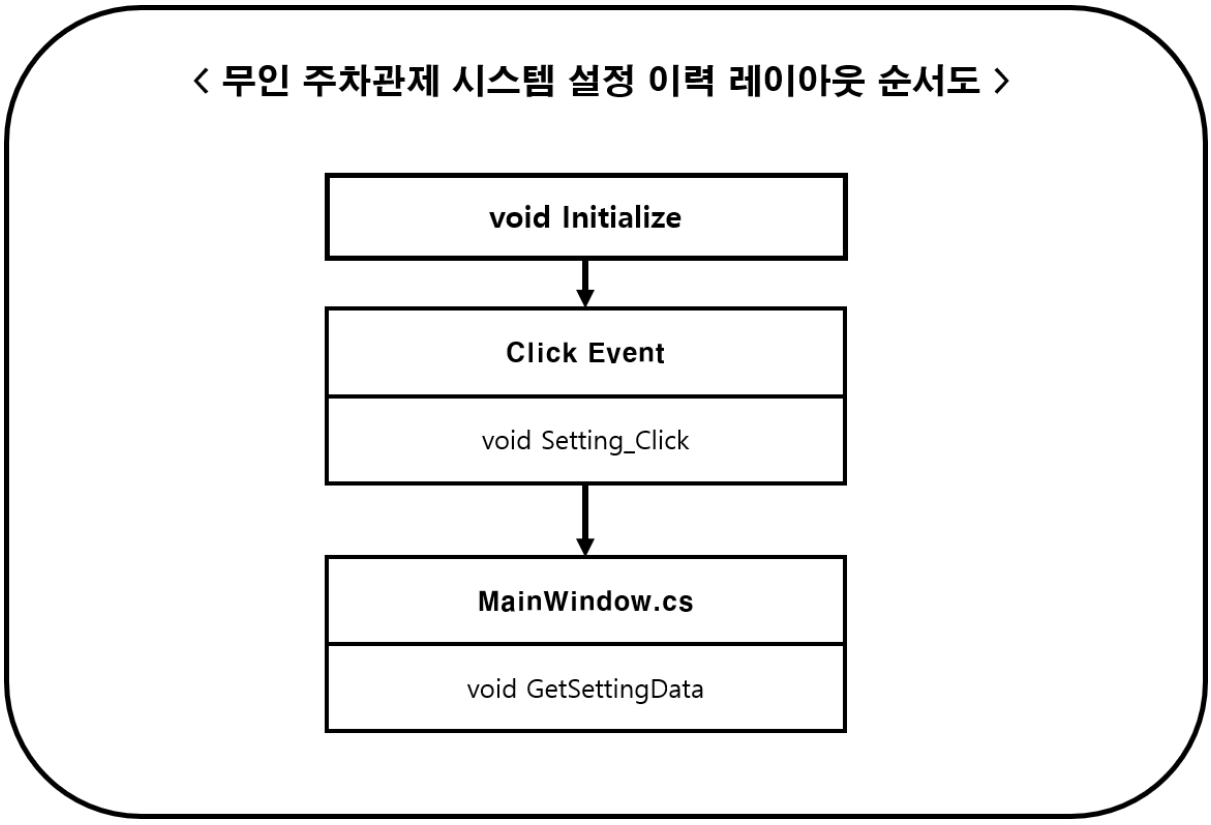

Figure 8. 무인 주차관제 시스템 설정 레이아웃 순서도

IV. 프로젝트 후기

1. 어려웠던 점

- **디자인 설계 :** 프로그램을 제작하면서 신경 쓰지 않았던 부분이 디자인 설계였었다. 프로그램의 기능들을 먼저 프로그래밍하고 디자인 하였다. 그러나 실행 화면의 크기와 UI 배치, 이미지 등을 파워포인트로 제작하였는데, 계속해서 수정할 때마다 코드들을 추가하다 보니 소스가 매우 복잡 해졌다.
- **1인 개발** : 프로젝트 진행을 홀로 담당하다 보니 디자인과 설계할 시간이 부족했다. 그리고 오류를 발생하는 경우를 전부 찾기가 어려웠으며 지금까지 구현된 프로그램에는 많은 오류가 있다. 임시로 try-catch 문으로 예외 처리를 해 놓았지만 해결해야 할 부분이다.
- **쓰레드** : 시뮬레이션을 개발하면서 쓰레드에 대한 기본 개념이 잡히지 않은 상태였다. 코딩을 하면서 쓰레드 충돌과 동기화 문제도 발생하였다. 동기화 문제를 해결하기 위해서 싱글톤 코딩을 하였고 쓰레드 충돌부분은 쓰레드 간 인터럽트하는 방법, 멀티쓰레드 등 공부하는 기회가 되었다.

2. 보완해야할 점

- **오류** : 현재까지 발견된 오류들은 시뮬레이션이 동작하는 중에 차량 생성 대수를 설정할 시 발생하는 런타임 오류와 2층에 주차된 차량 출력 오류가 있다. 예외 처리와 조건문을 이용해 정상적으로 동작할 수 있도록 오류를 해결해 나갈 방향이다.
- **이벤트 추가 :** 시뮬레이션 이벤트에는 입차와 출차가 있다. 처음 계획했던 이벤트는 상황 부여도 추가하여 시스템의 성능을 향상시키고자 하였다. 상황 부여에는 잘못된 데이터가 들어올 시 사용자에게 선택지를 부여한다. 데이터를 수동으로 수정할지, 삭제할지의 선택지가 있다. 그리고 실제 일어날 수 있는 접촉사고 이벤트 등이 있었지만 입차와 출차 이벤트를 구현하는 데에 생각보다 시간을 많이 소비하여 구현하지 못했다. 프로그램을 업데이트를 한다면 이벤트를 추가로 넣어 시스템의 성능을 향상 시키고 싶다.
- **차량 정보 :** 주차 이력에는 입/출차한 차량에 대한 정보들을 출력한다. 하지만 현재 출력할 수 있는 정보는 적다. 차량 정보를 추가한다면 총 방문 횟수와 결제 방식에 따른 할인요금제 여부가 있다. 그리고 배달 차량은 무료 요금으로 구현해 놓았지만 승용차 중 무료주차 권한부여 기능 등 추가하여 사용자가 원하는 정보를 얻을 수 있도록 보완할 방향이다.