**C++프로그래밍**

**프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *Snake-game* |
| 팀 명 | *3조* |
| 문서 제목 | 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 0 |
| **Date** | 19 |

|  |  |
| --- | --- |
| **팀원** | 최진우 (팀장) |
| 노윤수 |
| 전홍선 |
|  |
|  |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 C++프로그래밍 수강 학생 중 프로젝트 “Snake-game” 를 수행하는 팀 “3조”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “3조”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 최종보고서-3조-Snake-game.doc |
| **원안작성자** | 최진우, 노윤수, 전홍선 |
| **수정작업자** | 최진우, 노윤수, 전홍선 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2023-06-19 | 최진우 | 1.0 | 최초 작성 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**목 차**

[1 개요 4](#_Toc43103653)

[2 개발 내용 및 결과물 5](#_Toc43103654)

[2.1 목표 5](#_Toc43103655)

[2.2 개발 내용 및 결과물 6](#_Toc43103656)

[2.2.1 개발 내용 6](#_Toc43103657)

[2.2.2 시스템 구조 및 설계도 6](#_Toc43103658)

[2.2.3 활용/개발된 기술 6](#_Toc43103659)

[2.2.4 현실적 제한 요소 및 그 해결 방안 6](#_Toc43103660)

[2.2.5 결과물 목록 7](#_Toc43103661)

[3 자기평가 8](#_Toc43103662)

[4 참고 문헌 8](#_Toc43103663)

[5 부록 8](#_Toc43103664)

[5.1 사용자 매뉴얼 8](#_Toc43103665)

[5.2 설치 방법 8](#_Toc43103666)

# 개요

|  |
| --- |
| **평가기준 (10점)**  **프로젝트를 완성하기 위해 사용한 개발 방법을 기술하세요.**  **또한 사용하고 있는 외부 라이브러리와 해당 라이브러리를 획득/설치하는 방법을 기술하세요.** |

**해당 프로젝트는 ‘ncurses’ 라이브러리를 통해 TUI 기반의 Snake-game를 구현하는 프로젝트 입니다. 이를 위해 개발 OS에 따라 ‘brew’ 혹은 ‘apt’ repository를 통해 빌드 및 실행에 필요한 GCC/G++, make, cmake, ncurses 라이브러리를 획득하여 사용합니다.**

**해당 프로젝트를 진행하기 위해 Git을 사용하여 분산 작업환경을 구축하였으며, Branch 관리를 통해 과제수행 과정 및 작업공간 분리를 달성합니다.**

**해당 프로젝트를 보다 원활히 빌드하기 위해 빌드 자동화 도구인 ‘cmake’를 사용, Mac OS와 Linux(Ubuntu) 간 ‘ncurses’ 라이브러리를 불러오는 과정에서의 오류를 줄였습니다. 하지만 Windows의 경우 해당 목표를 달성하지 못하였습니다.**

**추가로 C++20 표준의 Module 기능을 사용하는 것을 최초에 목표하였으나 이를 지원하는 컴파일러가 Repository에서 구하기 힘든 점, 공통적으로 사용하는 IDE인 vscode에서 제대로 인식하지 않는 점을 들어 Module 기능은 사용하지 않았습니다.**

**본 프로젝트는 각 Map, Snake 구조를 class로 구현하였으며, 또한 ncurses의 WINDOW 구조를 보다 사용하기 쉽게 구현한 Window class를 구현하였습니다.**

**이를 상속해 Map과 Snake를 인자로 가지는 MainWindow class를 제작하는 등, 전반적으로 객체지향 프로그래밍 기법을 사용하였습니다.**

# 개발 내용 및 결과물

## 목표

|  |
| --- |
| **작성요령 (10점)**  **프로젝트의 목표를 기술하세요. 각 단계별 목표를 구체적으로 쓰세요.** |

|  |  |  |
| --- | --- | --- |
| 적용단계 | 내용 | 적용 여부 |
| 1단계 | Map의 구현 | 적용 |
| 2단계 | Snake 표현 및 조작 | 적용 |
| 3단계 | Item 요소의 구현 | 적용 |
| 4단계 | Gate 요소의 구현 | 적용 |
| 5단계 | 점수 요소의 구현 | 적용 |

## 개발 내용 및 결과물

### 개발 내용

**본 프로젝트는 제시된 과제인 Snake-game에 대하여 주어진 규격에 맞춰 C++으로 구현한 프로그램입니다. 해당 프로그램을 구현하기 위해 다음과 같은 세부 과제별 구현을 설명합니다.**

1. **Map의 구현 : 해당 Map 구조와 필요한 게임데이터를 저장하기 위해 맵 기초데이터인 이차원 정수형 배열과 해당 배열의 사이즈를 받아와 동적할당된 배열에 복사하는 Constructor를 구현하였습니다. 이후 ncurses WINDOW를 쉽게 구현, 조작하기 위한 Window class를 제작, 이를 상속한 MainWindow class를 만들어 Map을 출력하는 printMap method를 만들었습니다. 해당 Method의 정상작동을 확인하기 위해 Snake class를 미리 제작하여 사용하였으며 이에 대한 설명은 다음 문단에서 실시합니다.**
2. **Snake의 구현 : SnakeHead와 SnakeBody를 저장하기 위해 각 std::pair<int, int> 및 vector<pair<int, int>> 구조를 선택하였습니다. 그리고 \*Map를 받아와 움직인 Snake를 그리는 draw method를 제작, 해당 method가 작동할 때마다 snake가 지정된 방향으로 한 칸씩 움직이는 구조를 채택하였습니다. 다음으로 usleep() 함수를 통해 1틱(0.5s) 딜레이를 주고, getch() 함수를 통해 받은 입력값을 Snake의 움직이는 방향을 결정하는 move method의 인자로 주어 호출합니다. 마지막으로 \*Map 를 받아 Snake의 실제 움직임을 구현하는 draw() 함수에서 움직인 방향이 벽 혹은 SnakeBody 인지 확인, 해당된다면 게임의 메인 루프를 빠져나가는 구현입니다.**
3. **tem 요소의 구현 : 우선 Item을 사용할 때의 로직을 먼저 구현하였습니다. SnakeHead의 이동방향을 계산하여 해당 Item에 접근한다면, Item의 종류에 맞게 Snake의 길이를 늘이거나, 줄이는 로직을 먼저 구현하였습니다. Snake의 이동 후 해당 Item은 Item의 좌표 목록 std::list<std::pair<int, int>> 로 구현된 Map.itemLoc에서 제거됩니다. Snake의 움직임을 처리하는 로직 이후, Item이 5초 후 생성, 5초 후 삭제되는 것을 구현하기 위해 우선, ‘Tick’ 이라는 시간 단위를 정의하여, 1틱당 0.5초로 계산합니다. 가장 마지막 아이템이 생성된 시간으로부터 10Tick이 지났고, Map.itemLoc의 원소가 3개 미만일 때 랜덤으로 좌표를 생성해, 랜덤으로 아이템 종류를 선택, 해당 아이템을 선정된 좌표에 생성합니다. 해당 좌표가 빈 공간이 아니라면 새로 좌표를 선택, 해당 좌표가 빈 공간인지 다시 검사하여 빈 공간에만 아이템을 생성하도록 조치합니다. 마지막으로 최종 Item 생성 시간을 수정합니다.**
4. **Gate 요소의 구현 : Gate의 경우, Wall에서만 생길 수 있는 제약이 있어 Wall의 좌표를 담은 List 중에서 랜덤으로 선택하는 편이 훨씬 효율적이라 생각하였습니다. 이 전제에 따라 Map class의 생성자에서 맵 기초데이터를 복사할 때, Wall의 좌표를 담은 list를 같이 생성합니다. 이후 조건에 따라 Gate 생성시 std::advance() 를 이용 랜덤으로 선택된 i번째, j번째 Wall 좌표를 std::list<std::pair<int, int>>Map::gateLoc의 0/1번째 item으로 저장합니다. 이후 Gate를 통과시 gate 통과중인지 확인하는 Flag와, 다른 Gate에서 시계방향으로 진행 가능한지 탐색 후 가장 빨리 진행가능한 위치로 SnakeHead를 위치합니다. 이후 SnakeBody는 SnakeHead를 따라가도록 프로그래밍 되었기에 SnakeBody에 추가로 처리할 이유는 없습니다. 모든 SnakeBody가 Gate를 통과한 시점에서 flag를 초기화하며, 일정 시간 이후 Gate를 제거합니다.**
5. **점수 요소의 구현 : 점수 요소 및 해당 라운드에서의 미션 표시창인 ScoreboardWindow, MissionWindow는 각 Window class를 상속받아 매 틱마다 refresh하도록 구현하였습니다. 해당 class의 구조는 팀장 최진우가 구현하였습니다. 또한 스테이지별 맵 데이터, Snake의 위치, 미션 목록 lambda 함수 Vector를 반환하는 구조도 팀장 최진우가 구현하였습니다. 이후 각 맵의 디자인을 팀원 노윤수가 구현하였으며, 마지막으로 미션의 완료를 체크하는 로직을 팀원 전홍선이 마무리 하였습니다. 미션은 각 lambda 함수가 모두 true를 반환할 경우 맵 클리어로 간주하도록 구현되었습니다.**

### 시스템 구조 및 설계도

**제시된 과제인 Snake-game을 구현하고, STL 및 ncurses를 쉽게 다루기 위한 Wrapper class와 관련된 코드를 기술합니다.**

1. **Wrapper class의 구현 : 게임화면, 스코어보드 및 게임목표 화면을 구현하기 위해 ncurses의 WINDOW를 사용합니다. 이 WINDOW를 다루는 Wrapper class인 Window class를 구현하였습니다. 해당 Window class는 WINDOW의 내부 크기와 시작 x,y좌표를 받아 해당 영역에 WINDOW를 생성합니다. 해당 코드는 팀장 최진우가 제작하였습니다.**
2. **Map class및 Snake class, MainWindow class 구현 : Map class는 게임 진행에 있어 필요한 2차원 배열 데이터필드와 Wall 좌표의 std::list, 이외 각종 flag가 구현되어 있습니다. 또한 Snake class는 SnakeHea와 SnakeBody, 이외 Snake의 길이와 이를 수정하는 Snake::grow(), Snake::smaller(), Map을 받아와 Map의 데이터필드를 수정하는 Snake::draw(Map \*map) 중 Snake의 표현 및 조작에 필요한 코드를 구현합니다. 또한 getch() 함수로 받아온 키 입력을 처리하는 Snake::move(char dir) 를 구현합니다. 마지막으로 MainWindow class는 Window class를 상속받아 Map pointer와 시작 x,y 좌표를 받아 MainWIndow::printMap() 호출시 Map의 데이터필드를 바탕으로 화면에 출력합니다. 여기까지 팀장 최진우가 제작하였습니다.**
3. **Item 요소의 구현 : Item의 구현과 Item 획득 시 몸이 길이가 길어지거나 짧아지는 구현, 화면 출력과 관련된 MainWindow::printMap() method의 수정을 팀원 전홍선이 담당하였습니다. 이후 맵의 빈 공간에 랜덤으로 아이템을 생성/삭제하는 구현 및 디버깅을 팀원 전홍선과 팀장 최진우가 공동으로 수행하였습니다.**
4. **Gate 요소의 구현 : Gate를 임의의 벽 위에 생성/삭제하는 로직 및 Snake가 Gate를 이용 이동할 때의 요소 검사 및 움직임 처리를 팀원 노윤수가 맡았습니다. 이후 해당 코드의 디버깅을 팀원 노윤수와 팀장 최진우가 맡았으며, 해당 코드의 중복코드 정리를 팀장 최진우가 맡았습니다. 이외 이전까지 작성된 코드의 중복정리 및 최적화 작업을 팀장 최진우가 맡았습니다.**
5. **점수 요소의 구현 : 각 라운드별 맵, Snake의 위치, 미션 목록을 불러오는 로직을 팀장 최진우가 구현하였으며, 미션 목록은 미션의 타이틀, 미션의 수치, 해당 미션을 완수하였는지 확인하는 lambda 함수로 구성됩니다. 이후 Map 및 Snake pointer를 이용 미션 완료를 확인하는 lambda 함수 구성을 팀원 전홍선이 하였습니다.**

### 활용/개발된 기술

**본 프로젝트를 수행하기 위해 ‘cmake’ 및 ‘make’ 빌드 프로그램을 사용하였으며, GCC 혹은 clang 컴파일러를 통해 빌드 및 링크를 수행합니다.**

**TUI 환경에서의 게임을 구현하기 위해 ‘ncurses’ 라이브러리를 사용하였으며, 게임의 각 구성요소는 ncurses 라이브러리의 ‘WINDOW’ 구조를 통해 화면 분리를 실시합니다. 또한 이 WINDOW를 프로젝트에 필요한 method로 wrapping한 class Window 를 제작, 이를 각 요소에 맞춰 상속하여 MainWindow class 등을 구성합니다.**

**각 class에서 Map 혹은 Snake의 길이 등을 수정할 수 있도록 method를 구성하였습니다.**

**마지막으로 Map class의 구성 중 Map의 데이터필드는 2차원 정수 배열을 동적할당한 형태이며, Snake의 SnakeHead와 SnakeBody는 각 std::pair<int, int>, std::vector<std::pair<int. int>> 로 이루어져 있습니다. 2개의 데이터필드를 가지고 있는 std::pair 를 각 x, y좌표로 사용하고, 길이가 가변적인 SnakeBody는 std::vector를 이용 메모리 최적화를 달성합니다.**

**이 외 Item 요소의 위치를 기억하기 위한 std::list 등, STL Container를 적극 활용합니다.**

**마지막으로 데이터필드가 2차원 배열임에 따라 Gate의 구현 중 다음 이동할 좌표를 검색할 때 index가 -1 이하 혹은 Map::sz 이상임을 고려해야 했습니다. 하지만 Map::nextMove 함수를 Inline function으로 구현, 현좌표와 움직임 방향 정보, Map::sz로 나머지 연산을 실행하여 -1 혹은 Map::sz 이상의 값이 각 Map::sz – 1, 0 으로 대치되도록 구현합니다. 이 연산을 통해 데이터필드의 크기를 고려해야 했던 Gate의 구현이 대폭 간결해졌습니다.**

### 현실적 제한 요소 및 그 해결 방안

1. **Ncurses 라이브러리의 사용상 불편 : ncurses 라이브러리는 C언어를 사용함을 전제로 구성된 라이브러리로, 상당 부분이 절차지향을 기반으로 구성되어 있습니다. 하지만 C++ 사용을 전제로 Window class를 구현함에 따라 추상화가 가능해졌으며, 최초 ncurses 설정 부분을 제외한 대부분의 ncurses 함수 호출을 추상화 하였습니다.**
2. **Map datafield 접근 시 Outbound of Array 우려 : 해당 부분은 Map의 크기를 이용 나머지 연산을 통해 해결하였습니다. 해당 해설은 활용/개발된 기술 부분에서 기술하였습니다.**
3. **Makefile 구성의 불편 : 이번 프로젝트를 구성함에 있어 여러 Class로 나뉘어 있는 상황, ncurses 라이브러리를 포함한 빌드 환경을 구성함에 따라 빌드시스템 구성을 하여야 함을 인지하였습니다. 하지만 기존 Make는 파일이 늘어난다면 일일이 빌드 스크립트를 수정해야 하는 상황이 생깁니다. 이 불편을 줄이기 위해 cmake를 도입하였으며, ncurses 및 Class 구현파일을 포함한 빌드가 가능하도록 빌드스크립트를 작성하였습니다.**

### 결과물 목록

* **main.cpp – Snake-game 프로젝트의 구동 및 각종 제어를 담당합니다.**
* **CMakeLists.txt – 이번 프로젝트에서 채용한 Cmake의 빌드스크립트 입니다.**
* **lib/AdditionalWindow.(h/cpp) – ScoreboardWindow, MissionWindow의 구현입니다.**
* **lib/Enums.h – 맵 데이터필드상 구현요소인 Wall, SnakeHead 등을 구분하는 ElementType enum 및 방향키를 지정한 DirectionKey enum의 구현입니다.**
* **lib/GetStage.(h/cpp) – 각 스테이지의 맵 데이터필드, Snake의 초기위치, 미션 목록을 저장, 반환하는 함수의 구현입니다.**
* **lib/Maps.(h/cpp) – Map의 데이터필드, 상호작용을 구현입니다.**
* **lib/Snake.(h/cpp) – Snake의 움직임, Snake와 Map의 상호작용의 구현입니다.**
* **lib/Window.(h/cpp) – ncurses 라이브러리의 WINDOW 구성요소의 Wrapper class 구현입니다.**
* **lib/CMakeLists.txt – lib 폴더를 담당하는 빌드스크립트 입니다. lib 폴더의 C++ 구현을 빌드, 오브젝트 파일로 링킹하는 부분을 담당합니다.**
* **images – README.md 폴더에 삽입되는 이미지가 담겨있습니다.**
* **README.md – 현재 branch의 구현요소 등을 담은 문서 파일입니다.**

# 자기평가

* **팀장 최진우: 이번 프로젝트를 진행하며 ncurses 라이브러리를 처음 접하다보니 팀원 분들이 이 라이브러리를 쉽게 사용할 수 있도록, 구현상 어려움이 적어지도록 객체지향 프로그래밍을 많이 활용하였습니다. 저도 Wrapper class를 구성하며 많은 문서를 찾아보며 협업이 잘 될 수 있도록 노력하였습니다.**
* **팀원 노윤수: 다른 팀원과 협업을 하는 것은 처음이라 다소 어색했지만, 다른 팀원의 코딩을 이해하고, 그에 맞춰 기능을 구현해나가는 새로운 경험을 할 수 있었습니다. 타과생으로서 처음으로 듣는 코딩 수업이었는데, 프로젝트 기능을 구현하며, 배웠던 내용들을 하나하나씩 검토하여 적용시켜, 한 학기동안 배운 내용을 잘 정리하며 마무리할 수 있는 좋은 경험이 되었습니다.**
* **팀원 전홍선: 알고리즘 문제를 푸는 것과 다르게 프로그램을 구축, 설계하는것은 굉장히 복잡하다는것을 느꼈습니다. 뼈대를 잘 잡아야 나중에 기능을 추가할때 어려움이 없지만 그 뼈대를 잡는일이 정말 어렵다는것을 알게되었습니다. 스네이크 게임이라는 비교적 간단한 게임을 제작하면서 이후에 더 복잡한 프로그램을 만들땐 이 프로젝트에서 얻은 경험을 잘 활용해서 더 나은 결과물을 만들수 있을것 같습니다.**

# 참고 문헌

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처 | 발행년도 | 저자 | 기타 |
| 1 | 문서 | KLDP Wiki – NCURSES Programming HOWTO | KLDP | 2014 | Gsong |  |
| 2 | 문서 | die.net – ncurses(3) – Linux man page | die.net | - | Zeyd M. 외4인 |  |
| 3 | 문서 | Cmake 할때 쪼오오금 도움이 되는 문서 | Gist doc | 2020 | graykode |  |
| 4 | 문서 | [cmake] 자동으로 컴파일 대상 파일 리스트 만들기 | Tistory kukuta | 2021 | kukuta |  |

# 부록

## 사용자 매뉴얼

**프로그램 실행 시 Snake-Game이 시작됩니다.**

## 설치 방법

**본 프로그램을 컴파일하기 위해서 make 및 cmake, g++, ncurses-dev 라이브러리가 필요합니다. 설치 방법은 각 OS마다 설명합니다.**

* **Mac OS: homebrew를 이용, ‘make’, ‘cmake’, ‘ncurses’ 를 설치합니다.**
* **Linux(Ubuntu): build-essential 패키지 및 ‘libncurses5-dev’, ‘libncursesw5-dev’ 를 설치합니다.**
* **Windows: 빌드 환경의 불완전성으로 지원하지 않습니다.**

**빌드시 ‘cmake .’ 명령어를 통해 makefile을 생성합니다.**

**이후 정상적으로 makefile이 생성됨을 확인한다면 ‘make’ 명령어를 통해 최종 프로그램을 생성합니다.**

**최종 프로그램의 파일명은 snake\_game.exe 입니다.**