# **C++프로그래밍**

**프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *Snake Game* |
| 팀 명 | *7 조* |
| 문서 제목 | 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 2.0 |
| **Date** | 2022-06-16 |

|  |  |
| --- | --- |
| **팀원** | 고승우(팀장) |
| 노종빈 |
| 조서진 |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 C++프로그래밍 수강 학생 중 프로젝트 “Snake Game”을 수행하는 팀 7조의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 7조의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 최종보고서-Snake Game.doc |
| **원안작성자** | 고승우, 노종빈, 조서진 |
| **수정작업자** | 고승우, 노종빈, 조서진 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2022-05-05 | 노종빈 | 1.0 | 최초 작성 | 맵 구현 |
| 2022-05-09 | 조서진 | 1.1 | Snake | 뱀 움직임을 구현 |
| 2022-05-16 | 노종빈 | 1.2 | Inc Item. &  Dec Item | 아이템 기능 추가 |
| 2022-05-31 | 고승우 | 1.3 | Gate & Score | Gate 와 mission 시스템 추가 |
| 2022-06-02 | 조서진 | 1.4 | Game over | Game over 시스템 보완 |
| 2022-06-15 | 조서진 | 1.5 | record | record 기능 추가 |
| 2022-06-16 | 모두 | 2.0 | 전체 | 보고서 전체 내용 정리 |

**본 양식은 소프트웨어학부 C++프로그래밍 과목의 프로젝트 결과보고서 작성을 위한 기본 양식입니다. 문서의 필수 항목을 제시하는 것이니 폰트, 문단 구조 등의 디자인 부분은 자유롭게 설정하기 바랍니다. 양식 내에 붉은 색으로 기술한 부분은 지우고 작성하기 바랍니다.**

**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.** |

팀원들과 협업하여 프로젝트를 수행하기 위해 GitHub을 적극적으로 활용했다. 프로젝트를 총 6단계로 나누고 3명의 팀원이 각자 2가지의 단계를 구현했다. 각 팀원이 어떤 부분을 맡을지는 랜덤으로 정했다. 서로의 코드를 쉽게 이해하기 위해서 주석도 상세히 적도록 했다. 각자가 맡은 부분의 구현이 끝날 때마다 화상회의를 통해 서로가 구현한 부분에 대해 설명하고 보완할 점에 대해 자연스럽게 의견을 나누며 프로젝트를 진행했다.

윈도우에서 개발을 진행하였기 때문에, 우분투의 ncurses를 대체하는 외부 라이브러리인 pdcurses를 사용하였다. 이를 설치하기 위해서 Visual Studio에서 라이브러리를 쉽게 관리하도록 도와주는 vcpkg를 활용하였다. Vcpkg를 설치하는 방법은 다음과 같다. 먼저 ‘git clone https://github.com/Microsoft/vcpkg.git’ 명령어를 통해 vcpkg 파일을 받아온 후, "bootstrap-vcpkg.bat"을 실행시켜 설치한다. 그리고 환경 변수에서 PATH를 설정한다. 이렇게 설치를 완료했다면 vcpkg를 이용하여 pdcurses 라이브러리를 쉽게 설치하고 Visual Studio에 연동시킬 수 있다.

# 개발 내용 및 결과물

## 목표

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

|  |  |  |  |
| --- | --- | --- | --- |
| 적용단계 | 내용 | 적용 여부 | 개발자 |
| 1단계 | Map의 구현 | 적용 | 노종빈 |
| 2단계 | Snake 표현 및 조작 | 적용 | 조서진 |
| 3단계 | Item 요소의 구현 | 적용 | 노종빈 |
| 4단계 | Gate 요소의 구현 | 적용 | 고승우 |
| 5단계 | 클리어 시 다음스테이지로 넘어갈 수 있는 Mission의 구현 | 적용 | 고승우 |
| 6단계 | 파일 입출력을 활용해 클리어 시간을 기록하여 Top 10의 기록을 볼 수 있도록 함. | 적용 | 조서진 |

**프로젝트의 목표:**

다양한 Class와 STL라이브러리 등을 활용해 Snake Game을 구현하여, C++ 언어에 한층 더 익숙해지고자 한다. 여러가지 소스코드를 잘 융합하여 프로그램을 제작해보고, 팀 프로젝트인 만큼 Github를 활용한 협업에도 익숙해지고자 한다. 또한 pdcurses라는 외부 라이브러리를 사용하기 때문에, 모르는 여러 기능을 찾기 위해 구글링을 해야 한다. 수많은 외부 라이브러리를 사용하는 법을 모두 숙달할 수는 없기 때문에, 적절한 구글링을 통해 원하는 정보를 얻는 능력을 기르는 것도 개발자에겐 필수적이다.

**1단계:** 게임의 맵(스테이지)을 구현하고, pdcurses 라이브러리를 활용하여 이를 화면에 출력한다.

**2단계**: Vector를 활용하여 Snake를 표현한다. 방향키를 입력받아 Snake의 방향을 설정시켜야 한다. 또한 Snake의 이동을 위한 알고리즘을 고안하며, Snake의 모습을 게임 화면에 일정 주기마다 업데이트 시켜야 하기 때문에 이에 관련된 코드도 작성한다.

**3단계**: item class를 생성하여 Growth item과 Poison Item을 Game Rule에 맞게 구현한다.

**4단계**: Gate 요소를 Game Rule에 맞게 구현한다.

**5단계**: 다음 스테이지로 넘어가기 위해 충족해야 할 Mission 요소를 구현한다.

**6단계**: 게임 클리어 시 클리어 시간을 기록하여 파일로 저장하고, 이를 활용해 Top 10 기록을 보여주는 새로운 창을 구현한다.

## 개발 내용 및 결과물

### 개발 내용

|  |
| --- |
| **작성요령 (10점)**  **프로젝트의 수행의 내용을 구체적으로 기술한다. 세부 목표별로 어떤 결과를 어떤 방법으로 달성하였는지를 자세히 기술한다.** |

**0단계:** 1단계 map 구현을 시작 하기 전, 프로그램이 어떠한 구조로 짜여질지 간단히 구성을 하였다. 회의를 통하여 record를 추가하기로 했기 때문에, 메인메뉴에서 start game, record, exit를 선택할 수 있게 만들었다. strart game을 고를 시 새로운 window를 생성해, 그곳에서 게임이 실행된다.

**1단계:** 가장 기본이 되는 map부터 구현을 하였다. 맵 요소를 int 형 또는 char형 둘중 하나로 선택을 할 수 있었는데, 다양한 문자가 가능한 char을 쓰기로 결정을 하였다. 각 map의 요소들을 어떤 ASCIIcode로 할지는, 개발 중 편의를 위하여 벽을 ‘1’,’2’로 설정하였으며, 아무 요소도 없어야 하는 공간은 플레이 및 테스트 시 편의를 위해 화면상 표시되지 않는 ‘ ‘ (공백문자)로 설정을 하였다.3개의 map은 mapList라는 하나의 배열안에 저장하여 관리한다. GetMap함수는 mvwprintw함수를 이용하여, 화면에 2차원 배열인map을 요소하나씩 불러와 화면에 불러와준다. 추후 맵에 변경이 생기면(item 생성, snake이동 등) 이 함수를 이용하여 화면에 출력을 시켜주면 된다. 추가적으로 플래이어의 편의를 위해 map의 요소들을 구별이 잘 되는 색깔들로 나타내었다.

**2단계:** snake를 이루는 머리 및 몸통 한 부분 부분에 대하여 그 x좌표와 y좌표를 저장할 수 있도록 Snakepart라는 클래스를 정의했다. 즉 snake는 vector<Snakepart>형식이다. 일정 시간마다 플레이어의 방향키를 입력받아 snake의 방향인 headDir을 설정한다. 매 화면 업데이트마다, snake를 이루는 Snakepart의 x,y좌표를 변경함으로써 snake의 이동을 구현했다.

**3단계:** 무작위 위치에 아이템을 생성하기 위하여, 수업시간에 다룬 random함수를 이용한다. 난수생성이 시간을 시리얼로 받아 출력되기 때문에, 랜덤성을 위하여 아이템을 생성할 때마다 새로운 시리얼넘버를 주었다. 맵의 가로 세로 사이즈를 받아온 뒤, map의 위치를 고르고, 그곳의 map의 요소가 ‘0’이라면 아이템을 만들어준다. 이때 어떤 아이템이 생기는 지도 랜덤이다. 아이템을 먹었을때 snake의 길이를 조절하기 위해, snake.cpp의 moveSnake부분을 수정하였다. 아이템을 먹었을때, snake vector에 요소를 하나 더 추가시켜준 뒤, 추가된 꼬리의 좌표를 먹기전의 마지막 꼬리의 좌표로 설정해주었다. 반대로 decrease item을 먹었을때, snake vector에서 마지막 요소를 받아와, 꼬리가 위치하고 있던 곳의 요소를 ‘0’ 으로 만들어 화면에 지운 뒤, snake vector에서 제거하여 더 이상 화면에 출력되지 않게 만들었다. ItemClass 를 만들어 주어 아이템 하나하나의 x,y좌표, 생성시간을 담아주었으며, 편리하게 관리하기위해 클래스 안에 아이템을 관리하는vector를 넣어주었다. vector안의 객체의 정보를 이용하여, 생성된지 5초가 지난 item을 map에서 지워주었다. stage가 넘어갈때, vector안의 요소들을 지워주어서, 메모리 관리를 한다.

**4단계**: Gate를 구현하기 위해서 Gate 클래스를 생성한다. Gate 클래스 멤버변수에는 각 Gate의 y 좌표를 나타낼 int y, Gate의 x 좌표를 나타낼 int x, 반대편 게이트 객체를 가리킬 포인터 Gate \* pairPtr, 각 게이트마다 나갈 수 있는 방향을 저장하기 위한 int exitDirect가 있다. 멤버 함수로는 각 x, y 혹은 둘 다 int로 받아들이고 그 값을 멤버 변수에 저장하는 setX, setY, setPos함수, 각 값을 리턴하는 getY,getX 함수, Gate의 쌍을 지정하는 포인터 값을 입력 받아 멤버 변수에 할당하는 setPairPTr함수, 해당 값을 출력하는 getPairPtr함수 그리고 snake가 gate에서 나올 때를 위해 리턴할 x,y값을 위해 getExitY, getExitX 함수가 존재한다. getExitY와 getExitX 함수 같은 경우 snake의 headDir에 따라 리턴할 값이 달라지기 때문에 extern을 이용해 headDir을 가져와 각 상황에 따라 알맞게 y와 x값을 리턴하고 headDir값도 변경을 한다. 각 gate는 스테이지에 존재하는 난수를 벽의 개수로 나눈 후, 왼쪽 위를 기준으로 시계방향으로 돌아가며 각 벽의 임의의 숫자를 할당하고 난수에 해당하는 벽을 gate로 치환하는 방식으로 gate를 만들었다.

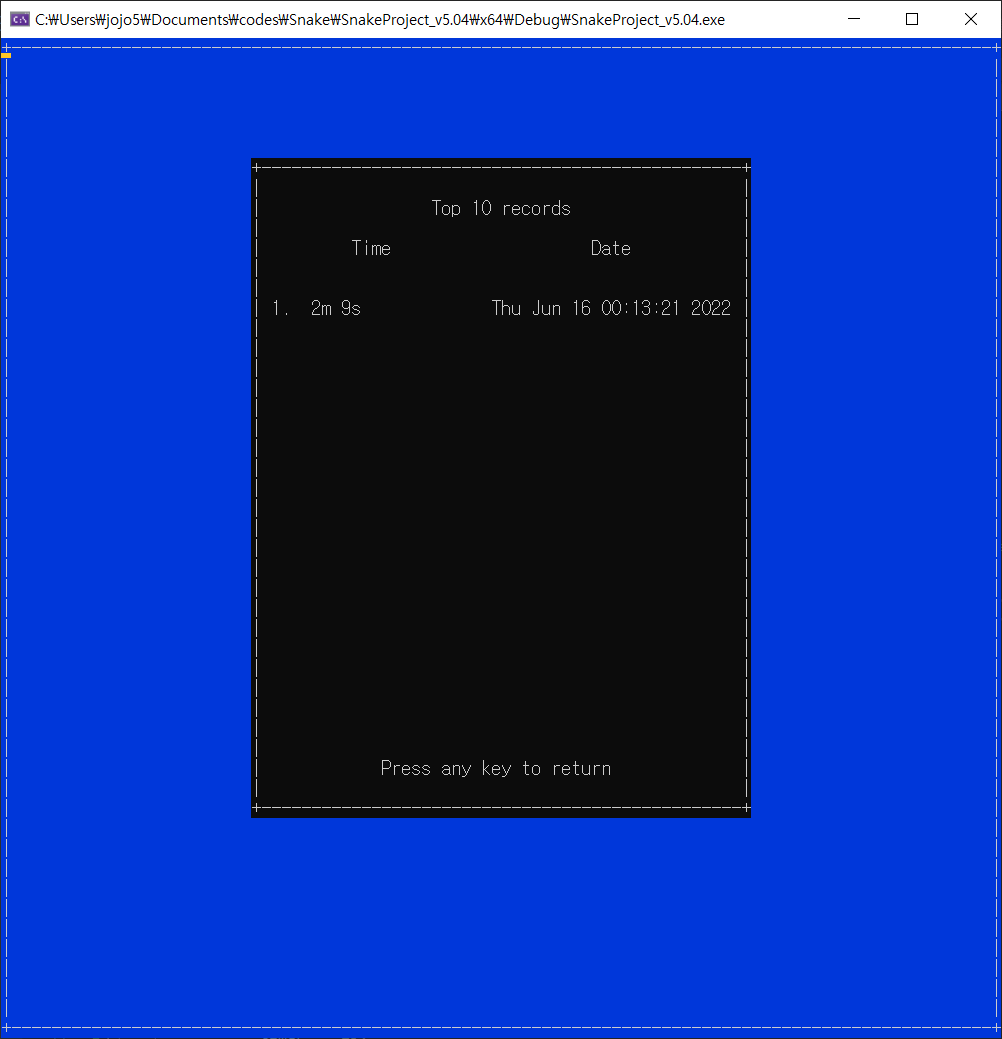
**5단계:** Score 클래스 멤버 변수에는 뱀의 길이를 저장할 int snake\_length, int inc item을 먹은횟수를 저장할 int num\_incItem, dec item을 먹은 횟수를 저장할 int num\_decItem, gate를 통과한 수를 저장할 int num\_gate, mission 성공 여부를 표현할 char check 그리고 clear 여부를 저장할 bool stageClear이 있다. 아이템을 먹거나 gate에 들어가는 것과 같은 이벤트가 발생했을 때, 각 해당하는 멤버변수의 값을 증가시키거나 감소시키는 함수를 호출한다. Check은 score board를 업데이트할 때마다, 각 조건이 성립이 됐을 경우 check에 ‘V’를 할당하여 출력했다. 또한 모든 조건이 true일 때 stageClear을 true로 반환한다.

**6단계**: record를 저장하기 위해 파일 입출력을 활용한다. setRec() 함수를 사용하여 ‘record.txt’에서 기록을 불러와 vector<vector<time\_t>>타입의 rec 변수에 저장한다. 만약 record.txt 파일이 없다면 새로 파일을 생성하고, 더미 값을 넣어준다. 이 더미 값은 게임 화면에 표시되진 않지만 형식상으로 rec에 저장되기는 한다. rec은 클리어 시간에 대해 오름차순 정렬한다.

플레이어가 게임을 클리어 했다면, 게임 플레이 날짜와 클리어 시간을 계산하여 insertRec()을 활용해 rec에 값을 push\_back해준다. 그 후, 클리어 시간과 해당 날짜가 적힌 record 윈도우를 보여준다. 또한 saveRec()을 이용하여 rec을 record.txt 파일에 저장한다. 이 때 rec의 길이가 10을 초과했다면 10등을 넘는 기록은 삭제된다.

게임 시작 시의 메인 화면에서도 메뉴에서 2번을 선택하면 record를 볼 수 있도록 했다. 이 상태에서 아무 키를 누르면 다시 메인 화면으로 돌아간다.

▼record 창의 모습



### 시스템 구조 및 설계도

|  |
| --- |
| **작성요령 (30점)**  **프로젝트의 각 세부 목표의 주요 기능(알고리즘 등)에 대해서 기술한다. 세부 목표별로 수정한 프로그램 소스 파일을 나열하고, 해당 파일에서 세부 목표를 달성하기 위해 작성한 클래스/함수에 대해 나열하고, 각 요소에 대해 간략한 설명을 작성한다. 또한 각 요소의 개발자를 명시한다.** |

**2.2.2.1 SnakePart 클래스 - 2단계**

**개발자: 조서진**

**- 멤버변수**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **타입** | **명칭** | **설명** |
| public | int | x | Snake를 이루는 몸체 각 부분의 x좌표 저장 |
| public | int | y | Snake를 이루는 몸체 각 부분의 y좌표 저장 |

- **멤버함수**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **반환타입** | **명칭** | **설명** |
| public |  | SnakePart(const int xpos, const int ypos) | 생성자 |
| public |  | SnakePart() | 생성자 |

**- 그 외 Snake 관련 함수**

|  |  |  |  |
| --- | --- | --- | --- |
| **반환타입** | **명칭** | **설명** | **개발자** |
| bool | moveSnake(const int stage) | Snake를 이루는 몸체의 x,y좌표를 변경하여 움직임 구현, 전진방향에 있는 요소 검사(요소에따라 추가적인 동작 실행 길이증가, 감소, gate이동 ). 반환값은 게임오버 여부 | 조서진,노종빈 |
| Vector<SnakePart> | makeSnake(const int stage) | Vector 형태의 snake를 생성하여 반환 | 조서진 |
| bool | setHeadDir() | 키를 입력받아 Snake의 방향 설정, 반환값은 게임 오버 여부 | 조서진 |

**- 그 외 Snake 관련 변수**

|  |  |  |
| --- | --- | --- |
| **타입** | **명칭** | **설명** |
| int | headDir | Snake의 이동 방향 |
| Int[][] | dir | 위,아래,오른쪽,왼쪽 방향 저장 - snake 이동 시 사용 |

**2.2.2.2 Game 클래스 - 전반적인 게임 동작**

**개발자: 조서진, 노종빈**

**- 멤버변수**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **타입** | **명칭** | **설명** |
| public | int | stage | 현재 스테이지 |
| public | bool | gameover | 게임오버 여부 |
| public | bool | stageCleared | 스테이지 클리어 여부 |
| public | time\_t | starttime | 게임 시작 때의 시간 |
| public | time\_t | endtime | 게임 클리어 때의 시간 |

- **멤버함수**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **접근제한자** | **반환타입** | **명칭** | **설명** | **개발자** |
| public |  | Game() | 생성자 | 노종빈 |
| public | void | launchGame() | 게임 시작 시 호출 | 노종빈 |
| public | void | updateScreen() | 일정 주기마다 화면 업데이트 | 노종빈,조서진 |
| public | void | nextStage() | 스테이지 클리어 시 호출 | 노종빈,조서진 |
| public | void | gameStatusCheck() | 게임오버, 스테이지 클리어 여부 검사 | 조서진 |
| public | void | gameCleared() | 게임 클리어 시 호출 | 조서진 |
| public | void | gameOvered() | 게임 오버 시 호출 | 조서진 |

**2.2.2.3 Screen 클래스 - 게임의 window 관리**

**개발자: 조서진,노종빈**

**- 멤버변수**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **타입** | **명칭** | **설명** |
| public | Window\* | menu | 메뉴 윈도우 |
| public | Window\* | record | 기록 윈도우 |
| public | Window\* | gameover | 게임오버 윈도 |
| private | char[][] | menuList | 메뉴 목록 |
| private | int | menuLength | 메뉴 길이 |

- **멤버함수**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **접근제한자** | **반환타입** | **명칭** | **설명** | **개발자** |
| public | void | loadHome() | 홈화면 불러오기 | 노종빈 |
| public | void | loadMenu() | 메뉴 창 불러오기 | 노종빈,조서진 |
| public | void | loadRecord(bool isgamecleared, time\_t cleartime = 0) | 기록 창 불러오기 | 조서진 |
| public | void | resetscreen() | 화면 초기화 | 조서진 |
| public | void | gameoverScreen() | 게임 오버 시 창 불러오기 | 조서진 |

**2.2.2.4 record.cpp - 6단계**

**개발자: 조서진**

**-변수**

|  |  |  |
| --- | --- | --- |
| **타입** | **명칭** | **설명** |
| vector<vector<time\_t>> | rec | 클리어 날짜와 클리어 시간을 함께 저장 |

**-함수**

|  |  |  |
| --- | --- | --- |
| **반환타입** | **명칭** | **설명** |
| void | setRec() | 파일로부터 데이터를 불러와서 rec에 저장 |
| void | void insertRec(time\_t date, time\_t gametime) | 클리어 날짜와 클리어 시간을 rec에 추가 |
| void | saveRec() | 파일에 rec을 저장 |
| bool | cmp(vector<time\_t>& v1, vector<time\_t>& v2) | 2차원 벡터의 정렬을 위해 사용 |

**2.2.2.5 Gate클래스 - 4단계**

**개발자: 고승우**

**- 멤버변수**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **타입** | **명칭** | **설명** |
| private | int | x | Gate 객체의 x좌표 저장 |
| private | int | y | Gate 객체의 y 좌표 저장 |
| private | Gate \* | pairPtr | 쌍이 되는 Gate 객체의 포인터 변수를 저장 |
| private | int | exitDirect | Gate에서 나갈 수 있는 방향을 저장 |

- **멤버함수**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **반환타입** | **명칭** | **설명** |
| public |  | Gate() | 생성자 |
| public |  | ~Gate() | 소멸자 |
| public | void | delGates(const int stage, Map map) | 게이트 소멸 후 다시 벽으로 바꾸는 작업 |
| public | void | setPairPtr(Gate \* pair) | Gate 객체 포인터를 저장하여 Gate쌍을 저장 |
| public | Gate \* | getPairPtr() const | Gate쌍의 포인터를 리턴 |
| public | void | setY(const int elem) | Gate 객체의 y좌표 설정 |
| public | void | setX(const int elem) | Gate 객체의 x좌표 설정 |
| public | void | setPos(const int elemY, const int elemX) | Gate 객체의 y, x좌표 설정 |
| public | void | SetExitDirect(const int elem) | Gate 객체의 출구방향 설정 |
| public | int | getY() const | Gate 객체의 y좌표 리턴 |
| public | int | getX() const | Gate 객체의 x좌표 리턴 |
| public | int | getExitY() const | 반대편 Gate로 snake가 나올때의 y좌표 설정 |
| public | int | getExitX() const | 반대편 Gate로 snake가 나올때의 x좌표 설정 |

**- 그 외 Gate 관련 함수**

|  |  |  |  |
| --- | --- | --- | --- |
| **반환타입** | **명칭** | **설명** | **개발자** |
| void | SetGate(const int stage) | stage을 받아 switch문을 활용하여 각 단계에 해당하는 함수를 호출 | 고승우, |
| void | GateInStage0(const int stage) | GetGate()가 stage=0일때 호출하는 함수 벽의 개수 보다 작은 난수를 입력 받고 해당하는 위치에 있는 벽을 게이트로 생성, 단 조건문을 통해 게이트가 바로 옆에서 생성되는 경우는 없도록 함 | 고승우 |
| void | GateInStage0(const int stage) | GetGate()가 stage=1일때 호출하는 함수 벽의 개수 보다 작은 난수를 입력 받고 해당하는 위치에 있는 벽을 게이트로 생성, 단 조건문을 통해 게이트가 바로 옆에서 생성되는 경우는 없도록 함 | 고승우 |
| void | GateInStage0(const int stage) | GetGate()가 stage=2일때 호출하는 함수 벽의 개수 보다 작은 난수를 입력 받고 해당하는 위치에 있는 벽을 게이트로 생성, 단 조건문을 통해 게이트가 바로 옆에서 생성되는 경우는 없도록 함 | 고승우 |

**- 그 외 Gate관련 변수**

|  |  |  |
| --- | --- | --- |
| **타입** | **명칭** | **설명** |
| int | num\_gate | Gate 객체를 통과한 횟 |
| int | headDir | 뱀의 머리방향을 저장하는 변수.  이를 바탕으로  Gate에서 뱀이 나올 때 머리의 방향을 설정하게 해준다. |
| static int | tempExitY, tempExitx | 게이트 통과 후 꼬리가 게이트를 통과하는지 확인하기 위해 게이트에 들어갈 경우 반대쪽 게이트에서 나오는 부분을 좌표를 저장, 게이트 통과 후 0으로 초기화 |

**2.2.2.6 Score클래스 - 5단계,6단계**

**개발자: 고승우**

**- 멤버변수**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **타입** | **명칭** | **설명** |
| private | int | snake\_length | 뱀의 길이를 저장 |
| private | int | num\_incItem | incItem을 먹은 갯수 저장 |
| private | int | num\_decItem | decItem을 먹은 갯수 저장 |
| private | int | num\_gate | Gate를 통과한 수 저장 |
| private | char | check | mission 성공 여부를 표현할 문자 -> 'V' or ' ' |
| private | bool | stageClear | clear 여부 저장 |

- **멤버함수**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **반환타입** | **명칭** | **설명** |
| public |  | Score() | 생성자 |
| public | void | resetScore() | 모든 점수와 정보를 초기화 |
| public | void | updateScore(WINDOW \* win2) | 멤버변수 최신화와 화면에 이러한 정보를 출력 |
| public | void | addSnakeLength() | 뱀의 길이 +1 |
| public | void | subSnakeLength() | 뱀의 길이 -1 |
| public | void | addIncItem() | 먹은 incItem +1 |
| public | void | addDecItem() | 먹은 decItem 개수 +1 |
| public | void | addGate() | 통과한 gate 개수 +1 |
| public | int | getDecItem() | 먹은 decItem 리턴 |
| public | bool | isClear(WINDOW\* win3) | 클리어 조건 만족을 했을 경우 true, 아닌 경우 false 출 |

**- 그 외 Gate 관련 함수**

|  |  |  |  |
| --- | --- | --- | --- |
| **반환타입** | **명칭** | **설명** | **개발자** |
| bool | gameover | 게임오버 여부를 저장하는 변수로 decItem을 2개 이상 먹었을 경우 true로 설정하여 게임을 종료하도록 한다. | 고승우 |

**2.2.2.7 ItemClass - 3단계**

**개발자 : 노종빈**

**-맴버변수**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **타입** | **명칭** | **설명** |
| public | int | x | 아이템의 x좌표 |
| public | int | y | 아이템의 y좌표 |
| public | int | createTime | 아이템이 생성된 시간 저장, |
| public | int | countItem | 화면에 출력되는 아이템개수 |
| public | vector<ItemClass> | listOfItem | 생성된 아이템들을 포함하는 vector |

- **멤버함수**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **반환타입** | **명칭** | **설명** |
| public |  | ItemClass(const int ypos=0, const int xpos=0) | 생성자 |
| public | void | void itemCreator (const int stage) | 랜덤하게 아이템을 생성, 아이템을 vector에 추가 |
| public | void | void itemDeleter(const int stage) | 생성 된 후 5초 뒤 아이템을 맵에서 지우는 함수. |

**2.2.2.8 Map - 1단계, 4단계**

**개발자 : 노종빈,고승우,조서진**

**-맴버변수**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **타입** | **명칭** | **설명** |
| public | char | mapList | 아이템의 x좌표 |

- **멤버함수**

|  |  |  |  |
| --- | --- | --- | --- |
| **반환타입** | **명칭** | **설명** | **개발자** |
| void | getMap(WINDOW\* window, const int stage) | MapList에서 맵을불러온다. 요소하나씩 불러와 검사를 하여 정해진 색깔로 화면에 출력된다. | 노종빈 |
| void | setMap(const int stage, const int x, const int y, const char k = 'n') | Snake가 이동할때 map의 요소를 바꿔주어, 화면에 snake가 이동하는것처럼 보이게한다. | 조서진 |
| void | setGate(const int stage); |  | 고승우 |
| void | gateInStage0(const int stage) |  | 고승우 |
| void | gateInStage1(const int stage) |  | 고승우 |
| void | gateInStage2(const int stage) |  | 고승우 |
| void | mapColorStart() | 플래이어의 가시성을 위하여, 각요소마다 색갈을 다르게 설정하기위해 만든 colorset | 노종빈 |

**2.2.2.9 main.cpp - 프로그램의 시작지점**

**개발자 : 노종빈,고승우,조서진**

|  |  |  |
| --- | --- | --- |
| **타입** | **명칭** | **설명** |
| int | main | 프로그램의 시작점. 실행시 mainMenu를 호출하고, 사용자에 선택에 따라 원하는 menu로 진입. Start game 선택 시 gameover가 됐는지 체크. gameover시 프로그램 종료 |

### 활용/개발된 기술

|  |
| --- |
| **작성요령 (10점)**  **프로젝트 수행에 사용한 외부 기술/라이브러리를 나열하여 작성한다. 각각 기술을 이 프로젝트에 적용할 때, 도움받거나 해결하고자 하는 기능에 대해 상세히 설명한다.**  **NCURSES / STL 라이브러리 등을 포함하여 설명한다.**  **또한, 이 프로젝트를 수행하면서, 새롭게 고안한 알고리즘 등이 있다면 설명한다.** |

**Vcpkg**: 윈도우 운영체제에서 외부 라이브러리를 쉽게 관리하도록 도와주는 vcpkg를 활용해 외부 라이브러리인 pdcurses를 쉽게 설치하였다.

**PdCurses** : ncurses와 비슷한 기능인 GUI를 지원한다. GUI는 사용자가 그래픽을 통해 사용자와 소통하는 방식을 말한다. GUI는 키보드와 마우스 모두 사용 가능하며, 그래픽으로 전달되는 그림이나 아이콘을 통해서 상호작용한다. GUI를 통해 다양한 입력 값을 받을 수 있고, 색상, 그래픽 등 다양한 그래픽을 출력할 수 있게 도와준다.

**GitHub(git)** : 서로가 작성한 소스파일을 쉽게 관리하기 위하여 github를 이용하였다. 특히 동시에 작업을 할 때 발생하는 merge문제를 git 덕분에 편하게 해결하였다.

**Vector**: Snake관리와 record 관리에 사용하였다. 동적 배열이기 때문에 길이가 계속해서 변하는 snake를 만드는 데 유용했다.

**Algorithm**: vector형식의 record를 sort하는 데 사용했다. record는 2차원 배열이기 때문에 2차원 배열의 두번째 원소를 기준으로 정렬하고자 비교 함수도 직접 만들었다.

**time.h:** 시간을 관리하기 위해 사용했다. 현재 날짜와 클리어 시간을 계산하는 데 사용했다.

**fstream**: record를 기록하기 위한 파일 입출력을 위해 사용했다.

### 현실적 제한 요소 및 그 해결 방안

|  |
| --- |
| **작성요령 (5점)**  **제안된 프로젝트의 단계 별 수행에 있어, 제한 요소를 찾아 작성한다. 해당 제한 요소를 해결하기 위해서 어떤 방법으로 해결하였는지 작성한다.** |

1단계: map을 만들때, unicode를 활용하여 특수기호로 다양한 요소들을 표현하려 했으나, 윈도우에서는 ncursesW가 사용이 불가해 포기하였다. 대신 pdcurses에 존재하는 색 변경 함수를 활용하여, 게이머가 편하게 요소들을 구별할 수 있게 프로그램을 만들었다

3단계: item을 각각 관리하여 map과 독립적으로 일정한 시간이 지나면 아예 메모리에서 없어지는 식으로 아이템을 구현하려했으나, 생성할 때, 사라질때, 신경쓸 것이 너무 많아, map과 연동하여 vector를 활용하여 관리하기로 하였다. map을 검사하여 item을 생성하고, item을 없엘때는 vector에 담긴 값을 활용해 map에서 지워주었다. map에서 사라진 item을 vector에서도 없에면 메모리 관리에 효율적이었을 것 같으나, item클래스가 크기가 작고, 그때그때 삭제 할때 버그가 생기는것을 고려해, stage가 바뀔때 item이 담긴 vector를 clear해주는 형태로 구성하였다. 해더파일과 cpp파일을 분리하고 싶었으나, 알수없는 에러가 발생하여 해더파일에 소스코드를 전부 구성해주었다.

4단계: 2개의 난수를 만들고, 만약 해당하는 좌표가 벽이라면 gate로 치환하는 방식으로 gate를 만들고 싶었지만, 극히 낮은 확률에 프로그램이 부드럽게 진행이 되지 않았다. 결국 각 스테이지 별로 벽에 임의로 숫자를 매긴 후, 해당하는 숫자의 벽을 gate로 치환하도록 하였고, 만약 두 난수의 차이가 1밖에 되지 않는다면, gate가 붙어서 생성될 수 있어 다시 난수를 돌리도록 예외를 두었다.

6단계: rec엔 time\_t타입이 저장되어 있는데, mvwprintw를 활용해 이를 윈도우에 출력시키려면 const char\* 타입으로 변환해야 했다. 먼저 time\_t타입을 long int 타입을 거쳐 string타입으로 변환하고, 이를 vector를 활용하여 vector<char>타입으로 다시 변환한다. 그 후 vector에 ‘\0’을 추가하고, vector 첫번째 원소의 값(&vector[0])을 mvwprintw의 const char\* 인자로 전달하여 윈도우에 출력시킨다.

### 결과물 목록

|  |
| --- |
| **작성요령 (5점)**  **결과물 목록을 작성한다. 목록은 제출하는 파일과 각 파일의 역할을 간략히 설명한다.** |

|  |  |
| --- | --- |
| **파일명** | **역할** |
| Game.cpp | Game.h를 활용하여 구현한 소스코드 |
| Game.h | 게임시작, 게임오버 및 화면 업데이트 등 게임의 중요 요소를 관리하는 소스코드 |
| gate.cpp | Gate.h를 활용하여 구현한 소스코드 |
| gate.h | 게임 요소 gate에 관한 소스코드 |
| Screen.cpp | Screen.h를 활용하여 구현한 소스코드 |
| Screen.h | 게임의 각종 window를 관리하는 소스코드 |
| ItemClass.cpp | ItemClass.h를 활용하여 구현한 소스코드 |
| ItemClass.h | 게임 요소 Item을 구현하는 소스코드 |
| Map.cpp | Map.h를 활용하여 구현한 소스코드 |
| Map.h | 게임의 Map(Stage)를 구현하는 소스코드 |
| Score.cpp | Score.h를 활용하여 구현한 소스코드 |
| Score.h | 게임의 Mission을 구현하는 소스코드 |
| snake.cpp | snake.h를 활용하여 구현한 소스코드 |
| snake.h | Snake의 이동 및 방향설정 등 snake 관련 기능을 구현한 소스코드 |
| record.cpp | 파일 입출력을 통해 기록을 저장하고 불러오는 기능을 구현한 소스코드 |
| main.cpp | 게임의 흐름을 구현한 main.cpp |
| curses.h | Pdcurses 소스코드 |
| panel.h | Pdcurses 소스코드 2 |

# 자기평가

|  |
| --- |
| **작성요령 (5점)**  **프로젝트를 수행한 자기 평가를 서술한다. 팀원 개개인의 자기 평가가 포함되어야 하며, 본인의 역할, 프로젝트 수행 시 어려운 점, 도움이 되었던 점, 이 프로젝트 운영에 개선이 필요하다고 생각하는 점을 충분히 서술한다.** |

**고승우:** 나의 역할은 gate의 구현과 score시스템의 구현이었다. 프로젝트 수행 시 아직은 C++ 언어가 익숙하지 않아. 링크 에러와 기본적인 문법 에러 등 다양한 에러를 겪어가면서 언어를 더 능숙하게 다루기 위해선 더 많은 경험이 필요하다고 느꼈다. 이번 프로젝트를 통해 팀원과의 화합이 굉장히 중요하다는 것을 느꼈다. 팀원들에게 적극적으로 도움을 요청하고 도움을 받을 수 있다는 것도 팀 프로젝트의 큰 장점이라는 것을 깨달았다. 이번 프로젝트 운영을 하면서 개선이 필요하다고 생각했던 부분은 없었다. 오히려 이번 프로젝트를 통해서 팀 프로젝트를 진행하는 방법에 대한 확신이 생겼다.

**노종빈:** map 구현과, item 부분을 맡았다. 시작부분을 맡았기 때문에, 구성 부분도 맡게되었다. 강의때 배웠던 부분을 최대한 활용하면서 구조를 짰지만, 개발을하면서 계획대로 되지않은 부분도 있고, 강의가 진행되면서 실력이 늘다보니, ‘그때 왜이렇게 만들었을까’라는 아쉬움이남는다. 하지만 팀원 모두 유쾌하게 성실히 프로젝트를 했기 때문에 매우 만족할만한 결과가 나온거같다. 더불어, 팀 프로젝트시 필요한 요소들(git활용, 주석 등)을 팀원들과 함께 알아보는 시간도 가져 수업내용 외에도 알아가는 부분이 많았다. 이제 수업이 마무리되며, 언어와 구조에 어느정도 지식이 쌓여서 다음프로젝트때는 아쉬움 없이 할 수 있을 것 같다. 프로젝트를 하는데 많이 쓰는 c++ 문법들이 수업 후반에 있어서, 활용을 많이 못 한것같아 아쉽다. (학사일정 때문에 어쩔수 없을 것 같다, 수업을 땡겨서 하고, 시험 몇 주전부터 수업시간에 프로젝트를 진행하게 해도 좋을것같다.)

**조서진:** snake와 record 기능의 구현을 맡았다. snake는 게임의 핵심이라고 말할 수 있기 때문에, 이를 구현하는 법을 고안하면서 자연스럽게 프로그래밍 실력도 향상된 것 같다. 또한 record 구현에는 시험범위에 들어있지 않아 소홀히 할 수도 있었던 파일 입출력을 활용하였기 때문에 더 유익했다. 하지만 프로그램을 완성하고 나니 코드의 구조 등 여러 부분에서 아쉬움이 남는다. 프로젝트를 진행하기 전 전체적인 구조를 어느 정도 짜 놨다고 생각했지만, 진행하면서 새로 추가된 기능과 파일때문에 깔끔하지 못한 구조가 된 것 같아 아쉽다. 한편 팀 프로젝트인 만큼 git을 활용한 협업을 경험할 수 있어서 좋았다. 팀원들과 협의하고 함께 프로젝트를 완성해 나가는 과정은 정말 유익하고 즐거웠다.

# 참고 문헌

**참고한 서적, 기사, 기술 문서, 웹페이지를 나열한다.:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처 | 발행년도 | 저자 | 기타 |
| 1 | 웹페이지 | [Tool] vcpkg 설치 및 사용법 | https://jungwoong.tistory.com/77 | 2020.05.03 |  |  |
|  |  |  |  |  |  |  |

# 부록

|  |
| --- |
| **작성요령 (15점)**  **프로젝트의 결과물을 사용하기 위한 방법에 대해서 작성하세요.** |

## 사용자 매뉴얼

**Git clone** [**https://github.com/jollidah/cpp\_project**](https://github.com/jollidah/cpp_project)

**혹은 직접 링크를 통해 파일을 가져올 수 있다.**

**Snake\_Game\cpp\_project\SnakeProject\_v5.04\x64\Debug\SnakeProject\_v5.04.exe**

**을 실행하면 된다.**

## 설치 방법