**C++프로그래밍**

**프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *ncurses을 이용한 Snake Game* |
| 팀 명 | *송태호* |
| 문서 제목 | SNAKE GAME 결과보고서 |
| Github | https://github.com/SongTaeHo1/SnakeGame |
| 실행 영상 | https://www.youtube.com/watch?v=C6VoZ0WVboM |

|  |  |
| --- | --- |
| **Version** | 1.0 |
| **Date** | 2022-Jun-15 |

|  |  |
| --- | --- |
| **팀원** | 송 태호(20181632) |

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

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | C++ 프로젝트 보고서 - 20181632 송태호.docx |
| **원안작성자** | 송태호 |
| **수정작업자** | 송태호 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2022-06-15 | 송태호 | 1.0 | 보고서 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 7](#_Toc43103656)

[2.2.1 개발 내용 7](#_Toc43103657)

[2.2.2 시스템 구조 및 설계도 8](#_Toc43103658)

[2.2.3 활용/개발된 기술 12](#_Toc43103659)

[2.2.4 현실적 제한 요소 및 그 해결 방안 13](#_Toc43103660)

[2.2.5 결과물 목록 13](#_Toc43103661)

[3 자기평가 14](#_Toc43103662)

[4 부록 15](#_Toc43103664)

[5.1 사용자 매뉴얼 15](#_Toc43103665)

[5.2 설치 방법 15](#_Toc43103666)

# 개요

|  |
| --- |
| **- 국민대학교 “C++ 프로그래밍” 수업에서 전체적으로 학습한 내용을 정리하고 활용하기 위해서 “Snake Game” 프로젝트를 진행하였다.**  **- 단계별로 맵을 구성하고, 맵을 이동하는 를 구현하고, Snake의 을 구현하고, 점수판과 다양한 스테이지를 제작하여 게임을 완성한다.**  **- 사용한 라이브러리**  **<ncurses> : 텍스트 사용자 인터페이스로, 이 프로젝트에서는 맵과 Snake, Item, 점수판을 시각화 사용하는데 이용하였다.**  **<thread> : 프로그램의 진행을 위해 업데이트 간격을 조절하는데 사용되었다.**  **<deque> : “deque”라는 자료구조를 제공해주는 라이브러리로, 맵에서 사용될 Item과 Snake의 좌표를 저장하고 관리하는데 이용하였다.**  **- 게임 룰**  **1. 사용자는 방향키를 이용하여, Snake의 이동방향을 제어할 수 있으며, Snake가 지나는 Item과 포탈을 통해서 주어진 미션을 통과하고, 다음 스테이지로 이동하며, 점수를 얻을 수 있다.**  **2. Snake는 벽에 부딪히거나, 길이가 3 미만이 되거나, 자신의 몸에 충돌하거나 진행방향과 역방향으로 조작된 경우, 죽게된다.**  **3. 파란색 GROWITEM을 먹으면, 몸길이가 1늘어나며, 빨간색 POISONITEM을 먹으면, 몸길이가 1 줄어들며, 이런 Item은 각각 최대 3개까지 나타난다.**  **4. 포탈은 한쌍으로 나타나면, 서로 겹치지지 않고, 한 포탈로 들어갈 경우, 다른 포탈로 나오게 된다.**  **5. Snake의 움직임과 업데이트, 키의 조작은 0.25초 간격으로 제어된다.**  **6. 죽을 경우, 총점은 총 살아남은 시간 + 최대 길이 + 총 얻은 GROWITEM과 POISONITEM + 포탈 출입 횟수로 계산된다.** |

# 개발 내용 및 결과물

## 목표

|  |  |  |
| --- | --- | --- |
| 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 |

|  |
| --- |
| **1단계 MAP의 구현**  **- 맵은 4개로 구성되며, 이는 Snake가 지날수 없는 벽과 Item이 출현하고 Snake가 지날 수 있는 빈공간으로 구성되며, 빈공간에는 Item과 Snake가 지날수 있고, 지날수 없는 벽은 포탈이 생성되는 벽과 그러지 못하는 꼭지점의 벽으로 구성된다.**  **2단계 Snake의 표현 및 조작**  **- Snake는 사용자의 방향키 입력에 따라 자신의 방향을 바꿀 수 있어야하며, 벽에 닿거나, 자신의 몸에 닿거나, 길이가 3미만이 되면 죽어야한다. 또한 Snake는 자신의 머리부터 주어진 방향으로 진행되며, 역방향으로 조작하게 되면 죽는다. 그리고 Snake는 벽을 제외한 공간으로 이동할 수 있고 0.25초의 시간간격으로 움직이게 된다.**  **3단계 Item 요소의 구현**  **- Item은 진행방향으로 길이를 늘려주는 GrowItem과 꼬리쪽부터 길이를 줄여주는 PoisonItem으로 구성되며, 각 Item은 게임 시작과 동시에 1개가 출현하게되며, 이후 랜덤한 시간 후에 최대 3개까지 랜덤한 시간간격으로 출현하게 된다. 또한 일정시간 동안 먹지 않은 Item은 사라지며 다른 위치에 나타난다. 이 Item들과 Snake는 서로의 위치에 출현하거나 겹치지지 않고 GrowItem으로 늘릴수 있는 최대 길이는 20이며, PoisonItem을 통해서 Snake의 길이가 3미만이 될 경우, Snake는 죽는다.**  **4단계 Gate 요소의 구현**  **- Gate는 두개가 한쌍으로 출현하게 된다. 이는 랜덤한 좌표에 벽이 위치한 곳에만 나타나며, 서로 겹치지 않는다. Gate에 Snake가 진입하는 동안은 이 Gate들은 사라지거나 다른 위치에 등장하지 않는다. Gate에 Snake가 통과할 경우, 다른 Gate부터 출현하게되며, 테두리에 있는 Gate에서 출현한 경우, 맵의 안쪽방향으로 진행방향을 갖게되며, 테두리에 있지 않을 경우, 자신의 원래 진행방향, 좌측, 아래 방향, 빈공간의 우선순위로 가능한 방향으로 진행하게 된다.**  **5단계 점수 요소의 구현**  **- 현재 스테이지의 게임정보를 표시해주는 점수판과 현재 스테이지의 미션을 제시해주고 달성 현황을 알려주는 미션판으로 구성된다. 또한 게임이 종료될 경우, 최종 점수와 게임종료를 알리는 문구가 출현하는 부분으로 구성된다.**  **- 4개로 구성된 맵은 각 맵에서 제시된 미션을 만족할 경우, 다음 맵으로 이동하게 된다. 이는 MapLevel을 통해서 구별하게 되며, 아무런 방해물도 존재하지 않는 처음 등장하는 기본 맵은 다시 등장하지 않으며, 방해물이 존재하는 스테이지가 계속 반복된다.**  **- 총점수는 총 살아남은 시간 + 최대 길이 + 총 얻은 GROWITEM과 POISONITEM + 포탈 출입 횟수로 계산된다.** |

## 개발 내용 및 결과물

### 개발 내용

|  |
| --- |
| **1. 맵의 구현**  **- 4개의 맵은 배열로 생성되며, 빈공간은 0, 포탈이 출현할수 있는 벽은 1, 포탈이 출현 못하는 immune wall은 2로 제시된다.**  **2. Snake의 구현**  **- Snake가 존재하는 좌표를 deque을 통해서 관리된다.**  **- 움직일 경우, 마지막 두개의 x와 y좌표를 pop하며, head에서 진행방향으로 존재하는 좌표를 push\_front한다.**  **- Snake가 존재하는 부분을 확인해주는 함수를 통해서 Item과 벽, 포탈에 대한 충돌이나 접촉 여부를 체크해주며, 맵을 그릴때 Snake의 위치를 파악하여 그려준다.**  **- deque의 push\_back과 pop\_back 함수를 통해서 꼬리 부분에 좌표를 추가하거나 줄임으로서 길이의 증가와 감소를 제어한다.**  **3. Item의 구현**  **- Item의 좌표를 저장해주는 deque를 생성하고, 이는 겹치지 않게 random한 좌표로 구성되며, 이 deque를 통해서 Item이 존재하는 여부를 확인해주는 함수를 만들며, 이를 통해서 map이 업데이트 될 떄 아이템의 위치를 파악하여 그려주고 Snake의 위치를 확인해주는 함수와 함께 사용되어 Item의 접촉여부를 파악해준다. Snake의 길이를 조절해주는 함수를 통해 아이템이 접촉될 경우, 조절한다. 또한 등장하는 시간과 제한은 랜덤한 시간간격으로 최대 3개의 제한을 지닌다.**  **4. Gate의 구현**  **-Gate의 좌표를 저장해주는 deque를 생성하고, 이는 겹치지 않게 random한 좌표로 구성되며, 한쌍이 존재한다. 한 Gate로 출입할 경우, 다른 Gate로 출현하게되며, 이러한 Gate는 배열에서 1의 값을 지닌곳에 랜덤하게 출현한다.**  **5. 전체적인 게임의 구현**  **- Snake 클래스의 isLife를 통해서 Snake의 사망여부를 판단하며 이는 게임이 업데이트 되며, 사용자의 입력을 받는 시간 간격인 0.25초마다 계속 진행된다. 또한 사용자의 입력을 받는 부분을 고려하여 nodelay와 noecho와 curs\_set(0)로 원활하게 해주며, 맵의 멤버 변수를 통해 총점의 계산되는 요소를 기록해주며, Snake 사망시, 이를 계산해준다.** |

### 시스템 구조 및 설계도

|  |
| --- |
| **1. Snake.h , Snake.cpp**  **1-1Member Variables**  **-char isDirection = Snake의 진행방향을 저장해주는 변수**  **-char prevDirection = Snake의 이전진행방향을 저장해주는 변수로 역방향 입력을 제어해준다.**  **-int isLength = Snake의 길이를 제어해주는 변수로 밑에 서술할 Snake의 좌표를 저장해주는 deque과 3미만이 될 경우를 판별해주는 역할을 한다.**  **-int isLife = Snake의 목숨으로 게임의 종료를 제어해주는 변수이다.**  **-deque<int> isBody = deque 자료구조로 Snake가 존재하는 영역의 좌표를 저장해준다.**  **1-2 Member Function**  **- void isDie(void) = 여러 사망 조건을 만족할 경우, 사용되는 함수로 게임의 종료를 위해 Snake 객체의 목숨을 0으로 초기화 해준다.**  **-void isMove(void) = 포탈이 아닐 경우, Snake를 움직여주는 함수로 현재 머리의 x,y좌표와 진행방향으로 x,y좌표를 변수로 갖고 만약, 자신의 몸에 충돌한다면 게임의 종료를 해주며, deque의 함수를 이용하여 진행방향을 push\_front()해주며 뒷부분을 pop\_back()해준다.**  **-void changeDirection(int key) = 사용자의 입력인 key를 인자로 받는 함수로 이를 통해서 Snake 객체의 진행방향을 위아래오른쪽아래를 순서대로 'u','d','r','l'로 제어해준다.**  **-void increaseL(void) = GrowItem을 먹을 경우, Snake의 길이가 최대길이인 20 미만이라면 사용될 함수로 진행방향으로 꼬리를 늘려주며, 길이를 나타내는 변수인 isLength 역시 늘려준다.**  **-void decreaseL(void) = PoisionItem을 먹을경우, Snake의 길이를 1 줄여주는 함수로 진행방향으로 꼬리를 하나 줄여주며, 길이를 나타내는 변수인 isLength 역시 줄여준다. 만약 이를 통해 길이가 3미만이 된다면 게임의 종료를 위해서 isDie()를 호출해준다.**  **-int getL(void), int getD(void) = 접근자로 Snake 객체의 길이와 진행방향을 각각 리턴해준다.**  **2. Map.h , Map.cpp**  **2-1 Member Variables**  **- Snake s = Snake 클래스의 객체로 Map을 이동해주는 Snake 객체이다.**  **- Int GrowItem , Int PoisonItem,Int PotalOn = GrowItem과 PoionItem의 갯수를 세주는 변수와 포탈이 존재하는지 여부를 판단해주는 변수이다. 이를 통해 아이템은 최대갯수까지 생성을, 포탈이 없을때 포탈을 생성해주는 영향을 끼친다.**  **-Int setPotal,Int anoRow,Int anoCol = 현재 포탈을 진행하여 다른 포탈의 좌표를 설정했는지를 판단해주는 변수와 이 판단된 다른 포탈의 행과 열을 저장해주는 변수이다.**  **- int plusNum, int minusNum,int GateNum,int maxLength = 현재 스테이지에서 목표를 달성하고 상태를 표시하는데 활용되는 변수이다.**  **- int totalPlus, int totalMinus,int totlaGate, int TotalMaxLength = 게임이 종료될떄 최종 점수 계산을 위해서 사용되는 변수로 모든 맵에서의 변수들의 합과 최대 길이로 기록된다.**  **- int startT, int timeS = 시간측정을 위한 변수로 게임이 시작될떄 초기화되는 startT를 통해서 현재 시간마다 업데이트되는 timeS와의 차로 시간을 기록하며 점수 계산에 이용된다.**  **- TotalScore = 총점을 기록하는 변수로 시간과 모든맵에서의 Item 획득 횟수의 합과 모든맵에서의 최대길이와 게이트 출입횟수의 합으로 계산된다.**  **-int targetL,int targetP,int targetM,int targetG = 랜덤의 값으로 프로그램이 실행될떄 한번 초기화 되는 변수로, 각 맵에서 달성해야하는 목표들의 수치를 기록하는 변수이다.**  **-deque<int> whereGrow,deque<int> wherePoison,deque<int> wherePotal = 각 GrowItem과 PoisonItem, Potal의 x,y좌표를 기록하는 deque이다.**  **- int mapLevel = 맵의 레벨로 초기화되는 변수**  **-int mapArray[40][50] = 현재 맵의 상태를 나타내주는 40\*50의 2차원 배열**  **-int mapStage[3][40][50] = 맵의 레벨이 변화할 경우, mapArray를 초기화해주기 위한 40\*50 짜리 방해물이 있는 맵이 3개 들어있는 3차원 배열**  **2-2 Member Function**  **- void CreateGrow(void) = 랜덤한 좌표에 현재GrowItem이 3개미만이라면 Poison Item이나 Snake의 위치와 겹치지 않는 벽이 아닌 위치에 GrowItem을 생성해주며 GrowItem 변수를 1 증가시키고 이 x,y좌표를 whereGrow에 저장해준다.**  **-void delGrow(int i, int j), void delGrow() = Snake가 Item을 먹은경우 사용되는 delGrow(int i, int j)는 먹은 좌표를 인자로 주고 해당 Item을 whereGrow에서 지우고 GrowItem의 갯수 역시 하나 지운다. 인자가 없는 delGrow()의 경우는 시간이 지났을때, 오래된 Item을 제거하는 목적의 함수로 가장 오래된 x,y좌표를 whereGrow에서 제거하고 GrowItem의 갯수를 하나 지운다.**  **-int isGrowHere(int i,int j) = 인자로 주어지는 행과 열의 값인 i,j를 whereGrow에서 검사해서 GrowItem이 있는 좌표값인 경우, 1을 리턴, 아닐경우 0을 리턴해준다.**  **- void CreatePoison(void) = 랜덤한 좌표에 현재PoisonItem이 3개미만이라면 Grow Item이나 Snake의 위치와 겹치지 않는 벽이 아닌 위치에 PoisonItem을 생성해주며 PoisonItem 변수를 1 증가시키고 이 x,y좌표를 wherePoison에 저장해준다.**  **-void delPoison(int i, int j), void dePoison(void) = Snake가 Item을 먹은경우 사용되는 delPoison(int i, int j)는 먹은 좌표를 인자로 주고 해당 Item을 wherePoison에서 지우고 PoisonItem의 갯수 역시 하나 지운다. 인자가 없는 delPoison()의 경우는 시간이 지났을때, 오래된 Item을 제거하는 목적의 함수로 가장 오래된 x,y좌표를 wherePoison에서 제거하고 PoisonItem의 갯수를 하나 지운다.**  **-int isPoisonHere(int i,int j) = 인자로 주어지는 행과 열의 값인 i,j를 wherePoison에서 검사해서 PoisonItem이 있는 좌표값인 경우, 1을 리턴, 아닐경우 0을 리턴해준다.**  **-void createPotal(void) = 현재 Snake가 포탈을 출입하지 않을때 실행되며, 현재 포탈의 좌표가 있는 wherePotal을 초기화해주며 서로 겹치지지않는 x,y좌표 두쌍을 포탈의 좌표로 랜덤하게 설정해준다. 이를 설정해주며, 포탈을 생성했다는 의미로 PotalOn을 1로 초기화해준다.**  **-int isPotalHere(int i, int j) = 인자로 주어진 x,y좌표인 i,j를 통해서 현재 포탈이 있는 좌표면 1을 아니면 0을 리턴해준다.**  **-void setGate(void) = 현재 들어가는 포탈을 판단하고 나오는 쪽의 포탈 좌표를 anoRow,anoCol에 초기화시켜주는 함수이다. 이를 통해 Potal을 설정해주었다는 의미로 setPotal 변수를 1로 초기화시켜주며 Potal 출입회수를 기록하는 변수인 gateNum과 totalGate를 1씩 증가시킨다.**  **-void PotalMove(void) = Snake가 Potal에 출입될때 사용되는 함수로 진행해야할 방향인 di를 변수로 갖는다. 테두리에 존재하는 게이트로 나올 경우는 맵의 안쪽 방향으로 1진행된쪽을 머리로 해서 방향역시 진출방향으로 설정하고 뒷부분을 제거하고 앞부분을 늘리는 형태로 이동하게되며, 테두리에 존재하지 않는 포탈의 경우, 4방향으로 자유로울 경우에는 진출방향 그대로 해서 방향을 설정해주고 진행방향으로 1진행된 쪽으로 진행하며, 상하로 자유로운지 좌우로 자유로운지 한방향만 자유로운지를 각기 검사해서 자유로운 쪽으로하여 원래 진행방향, 자유로운 방향으로 우선순위를 설정하여 진행방향 역시 초기화시켜주어 움직이게 된다.**  **-int isSnakeHere(int i, int j) = 인자로 주어진 x,y좌표인 i,j를 통해서 s.isBoby에서 x,y좌표를 확인하여 Snake가 존재하는 좌표라면 1을 아니라면 0을 리턴해준다.**  **-void isCrash(void) = Snake의 위치를 확인해주는 isSnakeHere()과 포탈의 위치를 확인해주는 isPotalHere()을 이용해서 현재 포탈이 아닌 벽에 충돌했는지를 검사해주고 충돌하였다면 isDie()를 호출하여 게임을 종료한다.**  **-void isItem(void) = Snake의 위치를 확인해주는 isSnakeHere()와 각 아이템의 위치를 확인해주는 isGrowHere(),isPoisionHere()을 통해서 아이템과 접촉했는지 검사해주며 접촉했다면 접촉한 x,y좌표를 통해 접촉한 아이템을 삭제하고 아이템의 효과를 적용하며 먹은 횟수를 기록하는 변수들의 값을 1씩 증가시킨다.**  **-void makeScore(void) = 검은 바탕에 흰 글씨의 속성을 적용하여 현재 길이, 최대길이, 현재 맵에서 얻은 GrowItem,PoisonItem,Gate출입횟수와 게임 동안의 실행시간과 최대길이를 표시해준다.**  **-void makeMission(void) = 검은 바탕에 흰글씨의 속성을 적용하여 현재 맵에서 얻어야할 목표치를 표기해주며 만약 성공했다면 (V)가 표기되게 시각화시켜준다.**  **-void initMap(void) = 처음 맵을 생성하기 위해서 사용되는 함수로 ncurses에서 제공되는 initscr을 통해서 윈도우를 생성하여 사이즈를 100\*100으로 재설정해주며, 커서와 입력반응을 안보이게 해주며, 아이템의 좌표가 저장된 deque을 초기화시켜주며, 맵과 Snake를 저장해준다.**  **-void mapLevelUp(void) = 미션을 클리어했을시 호출되는 함수로 방해물이 있는 맵들의 배열인 mapStage에서 각 맵레벨에 맞게 mapArray를 deepCopy해주며 각 아이템과 이들이 저장된 deque의 좌표들, 포탈과 포탈의 좌표들 맵에서 획득한 맵의 점수들을 0으로 전부 초기화시켜주며 Snake를 초기값으로 초기화시켜주며 맵의 레벨을1올려주고 만약 마지막 맵이라면 다시 첫번째 맵으로 되돌린다.**  **-void updateMap(void) = 매 0.25초마다 main.cpp에서 호출되는 함수로 변경된 정보들을 업데이트해서 맵을 다시 그려주며 Snake가 벽에 충돌했는지, Item을 얻었는지를 검사해주며, 점수판과 미션창을 갱신해주고, 미션을 달성했다면 맵의 레벨을 올리기위해서 mapLevelUp(void)를 호출해주며 만약 아이템이나 포탈이 하나도 없다면 최소한 하나씩 만들어준다.**  **3. main.cpp**  **3-1 Member Variable**  **-int randT,int randD,int randP = 아이템과 포탈의 랜덤한 생성주기를 위해서 만들어진 변수이다.**  **-Map m = 게임에 진행될 Map클래스의 변수이다.**  **3 - 2 진행**  **-사용자의 입력을 기다리지 않기위해서 nodelay를 설정해주며, 맵을 initMap()을 통해서 초기화시켜준다. 또한 rand()에서 난수 발생을 위해서 시드값을 변경하기위해서 srand(time(NULL))을 호출하며 Map m의 목표치를 랜덤하게 설정해준다.**  **- 반복문에서는 랜덤한 간격으로 아이템을 생성해주며 최대갯수라면 오래된 것을 지우고 새로운 것을 생성해준다.**  **- 또한 updateMap() 통해서 맵의 정보를 갱신하고 사용자의 입력을 받아들여 Snake의 진행방향을 설정해준다. Snake가 실패조건에 의해 죽는다면 게임을 종료하기위해서 반복문을 break해준다.**  **- 반복문을 나오게되면 최종점수를 계산해서 사용자에게 보여주며 최종 점수에 계산되는 요소들을 보여주고 키를 눌러서 게임을 종료하게 유도해준다.**  **- endwin()을 통해서 인터페이스를 완전히 종료하고 프로그램을 종료한다.** |

### 활용/개발된 기술

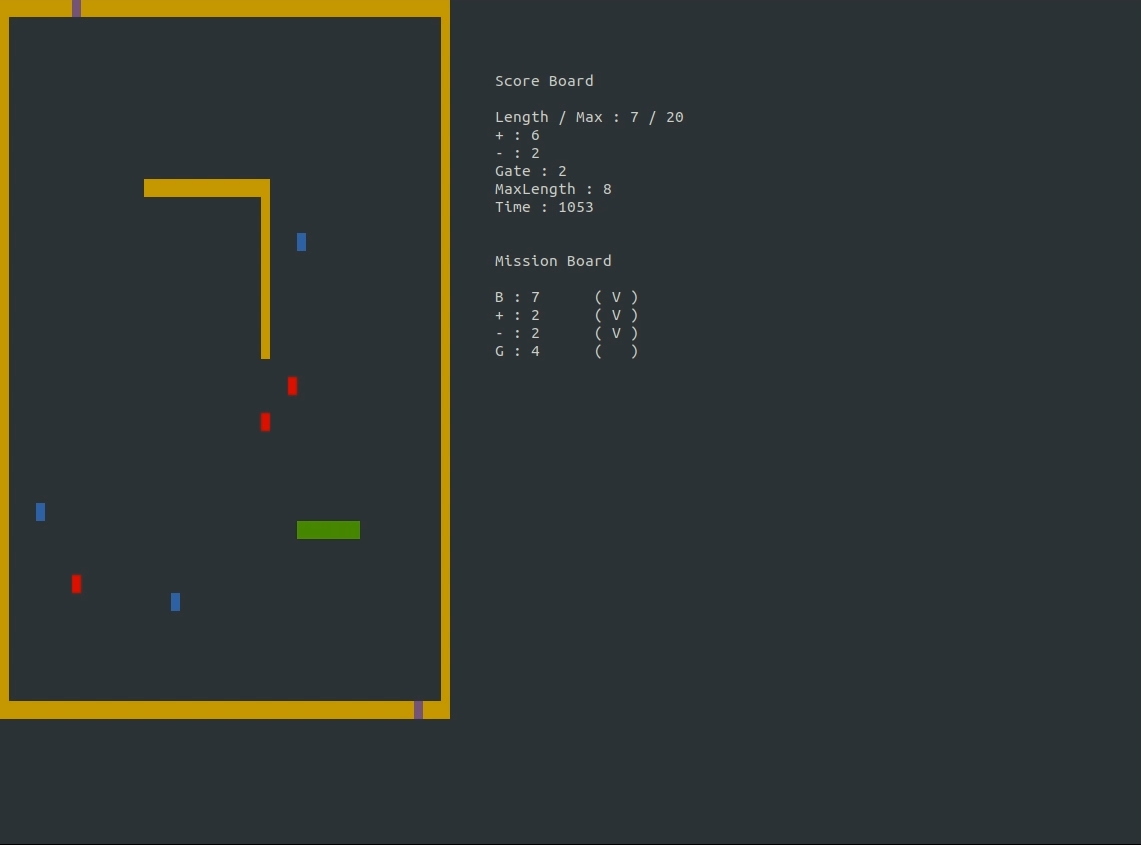
|  |
| --- |
| **<ncurses 라이브러리>**  **-텍스트 유저 인터페이스 라이브러리로 폰트의 색깔을 변경해주는 start\_color,init\_pair를 이용하였다.**  **- 또한 여러 윈도우와 마우스, 키보드에 관한 이벤트를 처리해주는 함수 역시 활용하였다.**  **<thread 라이브러리>**  **- this\_thread::sleep\_for(250ms)를 통해서 각 업데이트와 이벤트 처리에 대해서 250초의 시간간격을 주었다.**  **<deque 자료구조>**  **- Snake의 몸을 구현해주는 isBody , Item의 위치를 관리하는 whereGrow,wherePoison, 포탈의 위치를 관리하는 wherePotal에서 이용되었다.**  **- deque는 앞과 뒤에서의 push와 pop이 가능하고 랜덤접근이 가능한 자료구조이기에 이를 통해서 Snake의 길이와 이동, 아이템의 삭제와 생성, 포탈의 생성과 제거에 이용하였다.** |

### 현실적 제한 요소 및 그 해결 방안

|  |
| --- |
| **1. 사용자의 입력을 받는 getch() 함수에서 nodelay()를 이용해서 사용자의 입력을 기다리지 않아도 되지만 사용자의 입력이 처리시간보다 빠르게 연속된다면 처리에 늦는다.**  **1 - 1 해결 방법**  **리눅스의 경우 제공되지 않는 kbhit()의 함수를 통해서 사용자가 키보드 이벤트가 발생할때마다 key의 값을 초기화시켜주게 변경하여 이 문제가 해결가능하므로, kbhit()를 구현하면 해당 제한 요소를 해결할 수 있다.**  **2. 맵이 제작자가 설정하는 배열을 통해서 생성되기에 한계가 존재한다.**  **2 - 1 해결방법**  **맵의 기본이 되는 아무런 장애물이 존재하지 않는 베이스 맵을 40\*50의 2차원 배열로 초기화시킨뒤, 맵이 변경될떄마다 일정 확률로 몇개의 벽이 생성되는 함수를 생성하여 맵의 다양성을 증가시킨다.** |

### 결과물 목록

|  |  |
| --- | --- |
| 소스파일 | 설명 |
| main.cpp | 게임의 전체적인 흐름과 실행 |
| snake.cpp snake.h | Snake 객체의 이동, 죽음, 길이 제어 |
| map.cpp  map.h | 아이템, 포탈, 점수판, 미션창 업데이트 및 관리  맵의 효과, 충돌 감지 |
| makefile | 소스파일들의 컴파일 |

****

**<사진 1> 5단계의 결과물**

# 자기평가

|  |
| --- |
| **송태호(20181632) : 비대면 수업이라는 환경의 특수성과 여러 사항때문에 자동으로 구성된 프로젝트 팀원들과 연락이 되지 않아 프로젝트를 혼자 진행하게 되었다. 더군다나 C++과 주어진 ncurses 라이브러리를 처음 다루기에 많은 애로사항이 존재하였다. 적게는 2명에서 많게는 3명이서 진행할 프로젝트를 혼자 진행하게 된점이 가장 큰 문제점이라면 문제점 일 것이다. 그래도 많은 시간이 주어져서 많은 시행착오를 겪으며 5단계까지 전부 구현하였는데 이를 통해서 C++에 대한 이해도가 상승한 느낌이다. 또한 나는 수업을 들으며 deque등의 자료구조에 취약한 편이었는데 직접 여러 시도를 하면서 이에 대한 이해도가 늘어서 좋은 경험이 된 것 같다. 그리고 나는 스레드 등에 병렬 프로그래밍에 대한 개념이 부족하여 이를 완벽하게 프로그래밍 하지 못하고 전체적인 구조의 조잡합이 눈에 띄며 이러한 경험을 토대로 프로그래밍에서 설계 구조에 대한 중요성을 알게 되었다. 프로젝트를 통해서 얻은 점과 깨달은 부족한 점을 좀더 갈고 닦아야겠다고 느낀 좋은 경험이었다.** |

# 부록

## 사용자 매뉴얼

**- ncurses라이브러리가 없을 경우. 다음 코드를 터미널에서 실행하세요.**

**sudo apt-get update**

**sudo apt-get install libncurses5-dev libncursesw5-dev**

**1. ./snakeGame을 통해서 실행한다면 사용자는 방향키를 통해서 Snake의 방향을 조절할수 있습니다. 이 Snake는 0.25초마다 움직입니다.**

**2. 파란색은 GrowItem으로 먹을경우 길이가 1 늘어납니다. 최대 20까지 늘어납니다.**

**3. 빨간색은 PoisonItem으로 먹을 경우 길이가 1 줄어듭니다. 3미만이 될 경우, 죽습니다.**

**4. 보라색은 Potal입니다. 이 안으로 출입시 다른 Potal로 나오게 됩니다.**

**5. 노란색은 벽입니다. 부딪힐 경우 죽습니다.**

**6. 초록색은 Snake입니다. 자신의 몸에 부딪힐 경우, 죽습니다.**

**7. 오른쪽에 있는 미션의 4가지를 전부 깰 경우, 다음 맵으로 이동하게 됩니다.**

**8. 최대한 오래 살고, 많은 아이템을 얻어 최고점을 얻으시면 됩니다.**

## 설치 방법

**1. 소스 파일을 다운 받은 후, make snakeGame 명령어를 통해서 컴파일합니다.**

**2. ./snakeGame을 통해서 실행합니다.**

**- 실행 영상**

https://www.youtube.com/watch?v=C6VoZ0WVboM