



2020-2

C++프로그래밍 프로젝트

| 프로젝트명

Snake Game구현 및 맵 제작, 인공지능 대전 기능 구현

| 팀명

학습한 구렁이팀

| 문서명

결과 보고서

Version	1.4
Date	2020-06-20
지도교수	임은진
팀원	윤상건 (팀장)
	구형모
	김태윤



문서 정보

파일명	최종보고서- Snake Game구현 및 맵 제작, 인공지능 대전 기능 구현.docx
원안작성자	윤상건, 구형모, 김태운
수정작업자	윤상건, 구형모, 김태운

수정 내역

수정날짜	대표 수정자	Revision	추가/수정 항목	내 용
2020-06-16	윤상건	1.0	최초 작성	담당한 부분의 기술적 내용 작성
2016-06-18	김태운	1.1	내용 추가	담당한 부분의 기술적 내용 작성
2020-06-18	구형모	1.2	내용 추가	담당한 부분의 기술적 내용 작성
2020-06-19	윤상건	1.3	내용 추가	자기평가 추가
2020-06-20	김태운	1.4	내용 추가	자기평가 및 기술적 내용 보강

CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 C++프로그래밍 수강 학생 중 프로젝트 "Snake Game구현 및 맵 제작, 인공지능 대전 기능 구현"를 수행하는 팀 "학습한 구렁이"의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 "학습한 구렁이"의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.



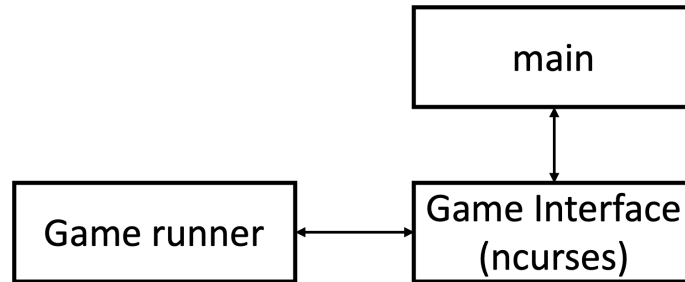
목차

1	개요.....	4
2	개발 내용 및 결과물.....	5
2.1	목표.....	5
2.2	개발 내용 및 결과물.....	7
2.2.1	개발 내용.....	7
2.2.2	시스템 구조 및 설계도.....	10
2.2.3	활용/개발된 기술.....	19
2.2.4	현실적 제한 요소 및 그 해결 방안.....	20
2.2.5	결과물 목록.....	21
3	자기평가.....	24
4	참고 문헌.....	26
5	부록.....	27
5.1	사용자 매뉴얼.....	27
5.2	설치 방법.....	30



1 개요

본 프로젝트는 Snake Game을 Ncurses 라이브러리를 사용하여 구현한 2020 국민대학교 C++ 강의의 기말 프로젝트로, 프로젝트의 간단한 구조는 다음과 같다.



Game runner는 Snake Game을 한 프레임 씩 사용자의 명령을 받아 작동하게 하고, 화면에 출력되어야 할 내용, 상태, 목표 달성 및 패배 여부를 판단 및 제공하는 역할을 한다. Game Interface에는 ncurses를 활용한 출력 부분이 들어간다. 게임 화면 이외에도 여러 메뉴 화면을 출력하는 등의 모든 입출력 부분을 담당한다. Snake game을 진행할 때에는 Game Interface가 Game runner의 객체를 생성하여 사용자로부터 받은 입력을 game runner에게 보내고 이를 바탕으로 game runner가 처리한 결과를 다시 game interface가 가져와 사용자에게 보여준다. main에서는 Snake game을 실행했을 때 이루어져야 하는 초기 작업들이 수행되고, Game Interface의 객체를 생성하여 게임 화면을 사용자에게 출력하기 시작하게 하는 역할을 한다.

각 파트 Game runner, Game Interface는 그 기능을 구현하기 위해 다시 여러 클래스로 세분화되었다. 각각의 파트 별 기능을 구현하기 위해 git을 활용한 개발 방법인 git-flow의 형태를 사용했다. 각자 맡은 기능에 대한 git branch를 만들고, 기능이 완성되면 코드 리뷰 과정을 거쳐 master branch로 merge되는 과정을 거쳐 개발했다.

프로젝트에서 요구하는 기초적인 Snake game을 개발한 뒤, 추가적인 기능인 맵 제작 기능과 Snake game을 플레이 하는 인공지능을 학습시켜 사용자와 대전을 하게 하는 기능을 구현했다.

개발 과정에서 사용한 외부 모듈은 총 1가지로, 콘솔 환경에서 입출력을 하기 위한 ncurses 라이브러리를 사용했다. 해당 라이브러리는 Ubuntu환경에서 다음의 명령어를 통해 설치할 수 있다.

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

구현된 프로젝트는 <https://github.com/c0510gy/Snake-Game> 에서 확인 할 수도 있다.



2 개발 내용 및 결과물

2.1 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
6단계	추가 기능 - 맵 에디터 구현	적용
7단계	추가 기능 - 인공지능 학습 및 인공지능과 대전 기능 구현	적용

위의 각 단계에 해당하는 자세한 구현 목표는 다음과 같다. 이때, 과제에서 제시된 1~5단계는 수정된 내용이 없다.

2.1.1 1단계 구현 목표

- NCurses Library 함수들을 사용하여 2 차원 배열로 된 Snake Map을 Game 화면으로 표시하는 프로그램을 완성한다. 그림의 세부사항은 각자 정한다.
 - Map은 21x21 을 최소 크기로 한다.
- 주의 : Wall 과 Immune Wall 을 잘 구분할 것

2.1.2 2단계 구현 목표

- 1단계의 맵 위에 Snake를 표시하고, 화살표를 입력 받아 Snake 가 움직이도록 프로그램을 완성한다.
 - Snake는 규칙 #1을 준수해야 한다.

2.1.3 3단계 구현 목표

- 2단계 프로그램에서, Map 위에 Growth Item과 Poison Item이 출현하도록 수정한다.
 - 게임 규칙 #2를 준수해야 한다.
- Growth Item과 Poison Item을 Map 배열에 표현할 때 값을 정한다
 - 화면상에 표현 시, 색이나 기호를 달리하여 구분할 수 있도록 한다
 - Map Data에서 Growth Item은 5, Poison Item은 6 과 같이 구분할 수 있도록 한다



2.1.4 4단계 구현 목표

- 3단계 프로그램에서, Map의 Wall의 임의의 위치에 한 쌍의 Gate가 출현할 수 있도록 변경하고, 각 Gate에 Snake가 통과할 수 있도록 수정한다.
 - 게임 규칙 #3, #4, #5를 준수해야한다
- Wall(Immune Wall 포함)과 Gate를 Map배열에 표현할 때 값을 결정한다
 - 화면상에 표현시, Gate는 Wall과 구분될 수 있도록한다
 - Map Data에서 Gate는 7과 같이 하여, 다른 요소와 구분할 수 있도록 한다

2.1.5 5단계 구현 목표

- 4단계 프로그램에서, 우측에 게임 점수를 표시하는 화면을 구성한다.
 - 게임 점수는 게임 규칙 #6을 준수한다
- Mission
 - 구성한 Map의 정의에 고정 값을 주거나
 - 매 게임마다 임의의 값을 주는 방식으로 처리한다
- Mission을 달성하면 다음 Map으로 진행하도록 프로그램을 완성한다
 - Stage는 최소 4개로 구성하고, 각 Stage의 Map은 서로 달라야 한다

2.1.6 6단계 구현 목표

- 사용자가 원하는 크기의 맵을 게임 내에서 생성 및 편집할 수 있도록 한다.
- Immune Wall은 사용자가 다루지 않고 자동적으로 처리된다.

2.1.7 7단계 구현 목표

- 구현한 Snake game을 학습한 인공지능을 구현한다.
 - 인공지능은 GA (Genetic Algorithm)과 MLP (Multilayer Perceptron)을 이용하여 구현 및 학습한다.
 - 학습 목표는 죽지 않고 최대한 많은 Growth 아이템을 획득 하는 것으로 한다.
- 학습된 인공지능과 사용자가 대전할 수 있는 기능을 구현한다
 - 최대 몸 길이를 점수로 하여 대결한다.
 - 사용자의 화면과 인공지능의 화면이 동시에 출력된다.



프로젝트 명	Snake Game구현 및 맵 제작, 인공지능 대전 기능 구현	
팀 명	학습한 구렁이	
Confidential Restricted	Version 1.4	2020-06-20

2.2 개발 내용 및 결과물

2.2.1 개발 내용

1단계 개발 내용

화면상에 진행중인 게임 맵을 출력하기 위해 게임 맵에 관한 정보가 담긴 MapItem클래스 객체를 생성자로 받는 GameRunner 클래스를 구현했다. GameRunner 클래스는 현재 맵 상태를 반환하는 메소드를 제공한다. 또한 맵과 사용자 파일을 불러오거나 저장하기 위한 FileManager 클래스를 구현하였다.

ncurses로 UI를 그리는 PlayerGameManager 클래스를 구현했다. GameRunner 클래스에서 맵상태를 받아온후 ncurses의 move() 와 printw()를 적절히 활용하여 맵을 화면상에 표시하였다.

2단계 개발 내용

Snake의 움직임을 구현하기 위해 Snake클래스를 구현했다. Snake 클래스에서는 이동 방향을 받고 해당 방향으로 움직이는 메소드를 제공한다.

두 가지 게임 종료 조건인 자기 자신과 충돌하는 경우와 벽과 충돌하는 두 가지 경우를 고려해 주기 위한 Snake를 관리하는 내용을 GameRunner 클래스를 구현했다.

기존 PlayerGameManager 클래스에서 벽 이외에도 Snake의 현재 위치를 받아와 화면에 표시 하도록 변경하였다. Ncurses 라이브러리의 getch() 를 통해 사용자의 입력을 받아왔으며, 이때 nodelay(stdscr, true) 함수를 호출하여 사용자 입력이 지속되지 않아도 딜레이 없이 게임이 진행되도록하였다. Keypad(stdscr, true)를 호출하여 방향키를 입력받도록 하였다. 딜레이 없이 게임이 진행된다면 매 프레임이 너무 빠르게 진행되므로 unistd.h 의 usleep()을 활용하여 프레임 속도를 조절하였다.

3단계 개발 내용

Snake 클래스가 Growth Item, Poison Item을 먹었을 각각의 경우를 처리해 주기 위해 각 경우에 대한 이동을 처리하는 메소드를 구현했다. 또한 Snake클래스 객체의 몸 길이가 3보다 작아졌는지 체크하는 로직을 추가했다.

GameRunner클래스에서는 Growth Item, Poison Item 각각을 지도상에서 업데이트 해주기 위한 메소드 두개를 구현했다. 이들은 각 프레임을 처리하는 단계에서 실행되며 각 아이템 마다 최대 3개까지 일정 확률로 지도에 추가한다. 랜덤한 위치에 Item이 나타나도록 아이템이 나타날 수 있는 위치 좌표를 std::set으로 관리해 주어 아이템이 발생한 장소는 랜덤 선택 후보에서 빠르게 제거할 수 있도록 했다. 또한 일정 프레임이 지나면 사용되지 않은 아이템을 제거하기 위해 std::queue를 이용하여 현재 존재하는 아이템들



프로젝트 명	Snake Game구현 및 맵 제작, 인공지능 대전 기능 구현	
팀 명	학습한 구렁이	
Confidential Restricted	Version 1.4	2020-06-20

을 시간순으로 관리하면서 가장 오래된 아이템 부터 존재 가능한 최대 프레임 수를 초과했는지 관리할 수 있도록 하여 아이템이 일정 시간이 지나면 사라지도록 했다.

기존 PlayerGameManager 클래스에서 벽, Snake 위치 이외에도 Growth Item과 Poison 아이템이 표시 될 수 있도록 변경 하였다. 부가적으로 화면상의 색을 활용하기 위해 attron(), attroff() 와 COLOR_PAIR(1) 등을 적절히 활용하여 각 아이템과 벽, Snake의 색상을 달리 하였다.

4단계 개발 내용

Snake클래스에서 snake객체가 Gate를 통해 다른 Gate로 이용하는 경우를 처리해 주기 위한 로직을 이동을 처리하는 메소드에 추가했다. 현재 Snake객체가 위치한 좌표에 Gate가 존재하면 다른 Gate의 좌표를 중심으로 규칙 4를 만족하도록 현재 Snake의 이동 방향, 시계방향, 반시계 방향, 반대 방향을 순차적으로 탐색하여 맵 내의 벽이 없는 장소로 이동 하도록 구현하였다. 규칙 3을 만족하기 위해서 현재 Snake가 아직 Gate를 통해 이동중인지 여부를 반환하는 메소드 또한 구현했다.

GameRunner 클래스에서 Gate의 위치를 맵 상에서 업데이트 해주기 위한 메소드를 구현했다. Item과 마찬가지로 Gate가 발생할 수 있는 좌표들을 관리해 주기 위한 컨테이너 객체를 두었다. 이때, Gate는 한 번에 오직 한개의 쌍만 발생하기 때문에 단순히 std::vector로 관리해 주었다. 관리되는 좌표들은 규칙 5를 만족시키기 위해 Immune Wall이 아닌 Wall들이 존재하는 좌표들이다. 또한 Item과 마찬가지로 일정 시간이 지나면 사라지게 하기 위해 마지막으로 Gate가 나타난 프레임을 저장해 주었다. 마찬가지로 Gate는 한 쌍만 발생하기 때문에 Item과는 다르게 정수형 변수 하나로만 관리해 주었다. 이때, Snake클래스 객체에서 현재 Snake가 Gate를 통과 중인지 여부를 가져와 규칙 3을 만족해 주었다.

기존 PlayerGameManager 클래스에서 벽, Snake 위치, Growth Item과 Poison 이외에도 Gate가 표시 될 수 있도록 변경 하였다. 또한 적절한 색과 문자를 이용하여 Gate와 Wall 을 직관적으로 구분하도록 하였다.

5단계 개발 내용

GameRunner 클래스에서 점수와 미션을 관리해 주기 위한 구조체 Score, Mission을 구현하고 이들을 통합적으로 관리해 주기 위한 클래스 StatusManager를 구현했다. StatusManager는 Score, Mission 클래스 객체 각각을 가지고 있고 점수 변동 사항을 수행하는 메소드를 제공한다. 이를 게임에 반영하기 위해 GameRunner에서 StatusManager클래스 객체를 멤버변수로 가지고 다음 프레임을 계산할 때마다 Snake클래스 객체에 넘겨주어 Snake객체의 상태에 따라 점수를 반영하도록 구현했다.



기존 PlayerGameManager 클래스에서 점수 판을 출력하도록 변경하였다. StatusManager 에서 Score 와 Mission을 받아오고 이를 newwin()을 통해 게임 판 옆에 출력하여 유저가 직관적으로 알수 있도록 구현했다. 또한, mission을 달성하면 다음 게임 판으로 넘어가도록 관리 해주는 GlobalStateManager 클래스를 작성하였다. 함수가 종료되면 호출했던 곳으로 돌아가는 특성을 활용해서 순차적으로 다음맵이 실행 되도록 구현하였다.

또한 사용자가 'p' 나 'q'를 누르면 언제든지 게임을 일시정지 하거나 끝낼 수 있는 기능을 추가 하였다.

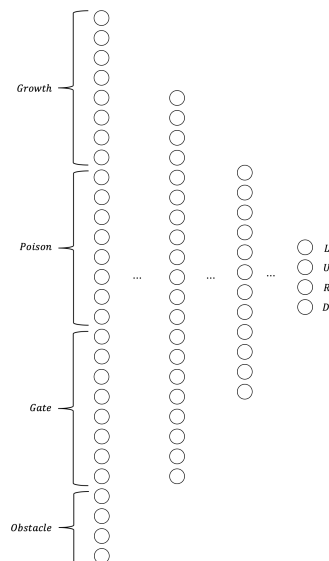
6단계 개발 내용

사용자가 입력한 너비와 높이에 맞는 맵을 생성해 편집할 수 있는 기능을 구현하기위해 MapEditor 클래스를 구현 하였다. MapEditor 클래스는 MapItem 객체를 생성하여 가지고 있으며, 사용자에게 입력 WINDOW를 보여주는 메소드, 입력한 높이와 너비에 맞는 초기 Map을 생성하여 화면에 보여주는 메소드, 키보드 입력에 맞추어 맵을 편집할 수 있는 메소드, 자동으로 Wall과 Immune Wall을 구분해 지정해주는 메소드를 제공한다. 사용자가 맵 편집을 완료하면, FileManager의 저장 메소드를 사용하여 .txt파일로 저장 한다.

맵 에디터, 커스텀 게임, 인공지능 대전을 사용자가 선택할 수 있도록 하는 메뉴 화면이 필요하게 되었다. 이에 MenuManager를 구현하여 직관적으로 사용자가 원하는 기능을 고를 수 있도록 하였다.

7단계 개발 내용

앞서 구현한 Snake Game을 플레이 하는 인공지능을 학습시키기 위해 GA(Genetic Algorithm)와 MLP(Multilayer Perceptron)을 사용했다. 먼저, 게임을 플레이 하는 MLP의 구조는 다음과 같다.





프로젝트 명	Snake Game구현 및 맵 제작, 인공지능 대전 기능 구현	
팀 명	학습한 구렁이	
Confidential Restricted	Version 1.4	2020-06-20

입력층은 총 28개의 퍼셉트론으로 이루어져 있으며, 각각 Growth Item, Poison Item, Gate의 각 방향에 따른 존재 여부와 각 방향에 따른 장애물로부터의 거리를 나타낸다. Growth Item, Poison Item, Gate의 경우 각각 Snake객체의 머리가 존재하는 좌표를 중심으로 8개의 방향에 대한 입력이 들어오며, 각 방향에 해당 아이템 또는 Gate가 존재하면 1로, 아니면 0으로 입력된다. 마지막 Obstacle의 경우 Snake객체의 머리가 존재하는 좌표를 중심으로 4개의 방향(상, 하, 좌, 우)에 대한 장애물(벽, Snake의 몸체)까지의 거리가 입력된다.

출력층은 총 4개의 퍼셉트론으로 이루어져 있으며, 각각 왼쪽, 위쪽, 오른쪽, 아래쪽의 다음 이동 방향을 뜻한다. 이 4개의 출력 값 중 가장 큰 값을 가지는 퍼셉트론의 방향을 다음 방향으로 한다.

해당 MLP에서 Activation function은 Sigmoid 함수를 사용했으며, 모든 각 가중치는 -1부터 1사이의 값이 되도록 했다.

학습은 GA를 이용하여 진행했다. 개체수를 2,000으로 두고 각각의 개체마다 3번 게임을 진행하여 그 중 가장 낮은 보상 함수 값을 가지도록 했다. 보상함수는 다음과 같이 주어지도록 했다.

$$fitness = \min(frames, 1000) \times (growth - poison)^3 + gate \times 2$$

이후 selection과정은 상위 50%에 대한 룰렛 휠 방식(Fitness proportionate selection aka. roulette wheel selection)을 사용했다. 마지막 crossover 과정에서는 선택된 두 개체의 보상 함수 값에 비례하여 랜덤으로 각 genome이 유전되도록 했다. 이 과정으로 하위 50%의 유전자를 대체했으며, 돌연변이 확률은 10%로 두었다.

학습 과정을 진행하기 위해 MLP, GA 클래스를 각각 구현했다. 이후 학습된 MLP가 게임을 각 프레임 단위로 실행한 결과를 받을 수 있도록 GameRunner 클래스를 상속받은 SnakeAI 클래스를 구현했다.

기존 PlayerGameManager 클래스를 변형하여 AiGameManager 클래스를 구현했다. 점수판 기능을 없애고 게임 화면을 두개 나란히 구성하여, 왼쪽에는 사용자의 게임 그리고 오른쪽에는 인공지능의 게임화면을 출력하도록 하였다. 게임의 종료 조건은 인공지능 보다 오래 살아남으면 승리, 아니면 실패 하는 방식으로 구현하였다.

2.2.2 시스템 구조 및 설계도

1단계 구현 내용

2차원 맵 정보를 관리하기 위해 단순 2중 for문을 활용하였다. 게임 맵을 관리하기 위해 key=value쌍의 데이터를 txt파일 형태로 Read/Write할 수 있는 클래스를 생성하여 활용하였다.

UI를 표현할 때에도 위 맵 정보를 마찬가지로 단순 2중 for문으로 읽어 들인후 switch-case 문으로 빈공간인지 벽인지 판단한후, ncurses의 move()와 printw()를 활용하여 화면상에 표현한다.



1단계 목표를 달성하기 위해 수정된 코드 내용은 다음과 같다.

수정한 파일 명	작성된 클래스/함수	간략 설명	개발자
include/Snake-Game/Primitives/Item.h	enum Item;	게임 맵에 저장될 각 항목들을 정의하기 위한 enum타입	윤상건
include/Snake-Game/Primitives/Point.h	struct Point;	게임 맵 등에서 2차원 좌표 값을 쉽게 다룰 수 있도록 하기 위한 구조체	윤상건
include/Snake-Game/Primitives/ MapItem.h	struct MapItem;	게임 맵에 관련된 모든 Item(이름, 생성일, 내용, 조건)등을 모두 담아 올인원으로 관리할 수 있도록 하기 위한 구조체	김태윤
include/Snake-Game/Primitives/MapManager.h	get(Point p) ; get(int x, int y);	get(point)의 형태로 받아 get(int, int) 함수를 호출하여 해당 좌표의 맵 상태를 반환	윤상건
	set(int x, int y, Item v); set(Point p, Item v);	set(Point, Item) 형태로 받아 set(int, int, item)함수를 호출하여 해당 좌표의 상태를 설정	윤상건
include/Snake-Game/Primitives/UserItem.h	struct UserItem;	사용자 정보를 저장하는 구조체	김태윤
include/Snake-Game/FileManager.h, src/FileManager.cpp	class FileManager;	txt파일로 저장되어있는 맵이나 유저 정보를 MapItem 또는 UserItem으로 객체로 반환해주거나 반대로 MapItem, UserItem을 txt파일로 저장하기 위한 클래스	김태윤
	bool isFileExist(std::string filePath);	file 존재여부를 확인	김태윤
	void writeMap(const MapItem& map, std::string filePath);	MapItem에서 정보를 읽어와 key=value 형태의	김태윤



		txt 파일로 지정된 경로에 저장	
	MapItem readMap(std::string filePath);	지정된 경로의 key=value 형태의 txt파일을 읽어와 MapItem으로 반환	김태윤
	void writeUser(const UserItem &user);	UserItem에서 정보를 읽어와 key=value 형태의 txt 파일로 지정된 경로에 저장	김태윤
	UserItem readUser(std::string filePath);	지정된 경로의 key=value 형태의 txt파일을 읽어와 UserItem으로 반환	김태윤
	void scanDir(std::string filePath);	지정된 경로의 파일 목록을 반환	김태윤
include/Snake-Game/Snake-Game.h src/Snake-Game.cpp	class GameRunner;	생성자에서 게임 맵 정보를 담은 MapItem클래스 객체를 받아 멤버변수로 저장해 게임을 진행하기 위한 클래스	윤상건
include/Snake-Game/PlayerGameManager.h src/PlayerGameManager.cpp	Class PlayerGameManager;	게임 화면을 그려주기 위해 만든 클래스	구형모
	void initializeWindow();	Ncurses 관련 초기화 및 설정을 하기 위한 클래스	구형모
	void play();	게임을 시작할때 불리는 함수 / 실질적으로 UI를 화면에 그려주는 곳	구형모

2단계 구현 내용

Snake의 이동을 효율적으로 처리하기 위해 double-ended queue자료구조를 활용했다. Snake의 이동은 맨 앞 (머리쪽)에 이동방향 좌표에 위치한 body를 새로 추가시켜 주고, 맨 뒤 (꼬리쪽)에 위치한 body를 제거해주는 방식으로 구현했다.

아이템 정보를 읽어들이는 반복문에서 switch-case 문에 snake도 추가 해서 화면상에 표시하도록 변경하였다.

화면이 맵보다 작으면 화면을 키우라고 사용자에게 알려주고 종료하는 로직을 추가 하였다.

2단계 목표를 달성하기 위해 수정된 코드 내용은 다음과 같다.



수정한 파일 명	작성된 클래스/함수	간략 설명	개발자
include/Snake-Game/Snake-Game.h src/Snake-Game.cpp	class Snake;	Snake를 처리하기 위한 클래스	윤상건
	Snake::getNewHead();	이동 방향에 대한 새로운 좌표(head) 가져오기	윤상건
	Snake::pushFront(Point body);	Head쪽에 새로운 몸통 (새로운 head) 추가	윤상건
	Snake::popBack();	꼬리쪽의 몸통 제거	윤상건
	Snake::checkValidity(MapManager& gameMap);	Snake 스스로 몸통이 겹치는지 검증	윤상건
	Snake::generalMove(Point newH);	위치 newH로 일반적인 이동	윤상건
	Snake::move(int direction, MapManager& gameMap);	direction방향으로 portals의 포탈을 가진 gameMap맵에서 이동연산 수행 유효하지 않은 상황이 발생할 경우 ERROR 반환	윤상건
	Snake::getHead();	현재 Snake의 머리(head)의 좌표를 반환	윤상건
	Snake::checkPoint(Point p);	위치 p가 Snake가 존재하는 위치에 속하는지 여부를 반환	윤상건
	getDirection();	현재 Snake의 방향 반환	윤상건
	GameRunner::nextFrame(int direction);	Snake가 direction방향으로 이동하는 다음 프레임을 계산하고 game over여부를 반환	윤상건
	GameRunner::getMap();	현재 진행중인 게임의 맵을 반환	윤상건
	GameRunner::getDirection();	현재 Snake의 진행중인 방향 반환	윤상건
include/Snake-Game/PlayerGameManager.h	void validateWindow();	화면 크기 체크 및 알림, 만약 너무 작으면 종료	구형모



src/PlayerGameManager.cpp

3단계 구현 내용

Snake의 Growth, Poison Item에 의한 효과를 구현하기 위해 double-ended queue를 활용했다. Growth의 경우 일반 이동 후에 deque의 back쪽에 새로운 body를 추가했고 Poison의 경우 일반 이동 후에 back쪽의 body를 제거했다.

Growth, Poison Item이 발생할 위치를 균일한 확률로 랜덤하게 선정하기 위해 선정된 좌표를 빠르게 제거할 수 있는 std::set 자료구조를 이용했다. 또한 오래된 아이템을 제거하기 위해 시간순으로 존재하는 아이템을 처리하도록 std::queue를 이용했다.

아이템 정보를 읽어들이는 반복문에서 switch-case 문에 Poison Item과 Growth Item 도 추가 해서 화면상에 표시하도록 변경 하였다. 또한 start_color() 와 use_default_colors()를 활용해 터미널에 컬러를 지원하도록 설정하였으며 init_pair(1, COLOR_WHITE, COLOR_WHITE) 를 미리 등록해두고 이를 attron(COLOR_PAIR(1)) 과 attroff(COLOR_PAIR(1))를 적절히 활용하여 사용자가 직관적으로 아이템, 벽, snake를 구분할 수 있도록 하였다.

3단계 목표를 달성하기 위해 수정된 코드 내용은 다음과 같다.

수정한 파일 명	작성된 클래스/함수	간략 설명	개발자
include/Snake-Game/Snake-Game.h src/Snake-Game.cpp	class RandomGenerator;	랜덤 숫자를 가져오기 위한 클래스	윤상건
	RandomGenerator::getRandom(int s, int e);	범위 [s, e]에서 균등 분포로 랜덤한 값 하나를 반환	윤상건
	Snake::growthMove(Point newH);	위치 newH로 성장 이동 - Growth 아이템 효과	윤상건
	Snake::poisonMove(Point newH);	위치 newH로 독 이동 - Poison 아이템 효과	윤상건
	Snake::move(int direction, MapManager& gameMap);	섭취한 아이템을 반환하도록 추가 수정	윤상건
	GameRunner::getRandomItemPoint (std::queue<std::pair<IndexedPoint, int>>& timeQ);	아이템 생성 후보 좌표에서 유효한 위치를 균등 랜덤으로 뽑아 반환 최대 100회 시도를 통해 유효한 위치를 뽑는데 실패했다면 {-1, -1}을 반환	윤상건
	GameRunner::updateGrowth();	Growth 아이템 업데이트	윤상건
	GameRunner::updatePoison();	Poison 아이템 업데이트	윤상건
include/Snake-Game/PlayerGameManager.h	void initializeColors();	터미널 상에서 색을 활용하기 위해 설정을 진행하는 함수	구형모



src/PlayerGameManag
er.cpp

4단계 구현 내용

Gate가 생성될 위치를 균일한 확률로 랜덤하게 선정하기 위해 후보 좌표 목록을 std::vector로 관리해 주었다. 아이템과 달리 vector로 관리한 이유는, Gate는 많아봐야 단 한 쌍만 존재하기 때문에 후보 좌표 목록에서 임의의 좌표를 삭제해 줄 경우가 없기 때문이다.

Snake객체가 Gate를 이용하는 경우를 처리하기 위해 move메소드에 Gate를 만난 경우, 반대쪽 Gate의 좌표를 가져와 해당 Gate에서 나오는 위치를 찾아 해당 좌표를 head로 두는 일반 이동을 하도록 구현했다. 또한 Snake가 Gate를 이용중인 경우를 체크하기 위해 Gate에 아직 나오지 않은 몸 길이를 이용하여 Gate를 이용중인지 여부를 반환하는 메소드를 구현했다.

아이템 정보를 읽어들이는 반복문에서 switch-case 문에 Gate 도 추가 해서 화면상에 표시하도록 변경하였다. 3단계에서 구현한 색 입히는 로직을 활용하여 Gate를 직관적으로 구분하도록 하였다.

4단계 목표를 달성하기 위해 수정된 코드 내용은 다음과 같다.

수정한 파일 명	작성된 클래스/함수	간략 설명	개발자
include/Snake- Game/Snake-Game.h src/Snake-Game.cpp	Snake::move(int direction, MapManager& gameMap);	Gate를 만났을 경우, 반대쪽 Gate의 나가는 좌표를 찾고 해당 좌표를 head로 하는 일반 이동을 수행하는 코드 추가	윤상건
	Snake::isInPortal();	현재 Snake가 Portal(Gate)를 타고 있는지 여부를 반환	윤상건
	GameRunner::updatePortal();	Portal(Gate) 업데이트	윤상건

5단계 구현 내용

점수와 미션을 관리하기 위한 클래스를 적용했다. 점수와 미션에 대한 정보를 각각 구조체 Score, Mission에 저장하고 StatusManager가 이 둘을 멤버변수로 가지며 관리한다.

점수판, 미션보드를 화면상에 표시하기 위해 newwin(), wrefresh(), mvwprintw() 를 활용하여 표시하였으며 점수, 미션 상황을 StatusManager의 Score, Mission 객체에서 받아오도록 하였다. 부가적으로 사용자가 언제든지 일시정지 및 게임을 나갈 수 있도록 pause 기능과 UI를 구현하였으며 이를 nodelay(stdscr, false), nodelay(stdscr, true)를 활용하여 일시정지와 재시작 기능을 구현하였다.

5단계 목표를 달성하기 위해 수정된 코드 내용은 다음과 같다.

수정한 파일 명	작성된 클래스/함수	간략 설명	개발자
include/Snake-	struct Score;	점수를 관리하기 위한	윤상건



Game/Primitives/Score.h		구조체	
include/Snake-Game/ Primitives/Mission.h	struct Mission;	미션 달성 여부를 관리하기 위한 구조체	윤상건
include/Snake- Game/StatusManager.h src/StatusManager.cpp	class StatusManager;	점수와 미션 달성 여부를 종합적으로 관리하는 클래스	윤상건
	StatusManager::scoreBody();	Snake의 몸 길이가 1 증가한 경우 점수 갱신	윤상건
	StatusManager::descoreBody();	Snake의 몸 길이가 1 감소한 경우 점수 갱신	윤상건
	StatusManager::scoreGrowth();	Snake가 Growth아이템을 획득한 경우 점수 갱신	윤상건
	StatusManager::scorePoison();	Snake가 Poison아이템을 획득한 경우 점수 갱신	윤상건
	StatusManager::scoreGate();	Snake가 Gate를 이용한 경우 점수 갱신	윤상건
include/Snake-Game/Snake- Game.h src/Snake-Game.cpp	Snake::move(int direction, MapManager& gameMap, StatusManager& status, std::vector<Point>& portals);	StatusManager객체를 전달받아 Snake이동 시 점수 반영하도록 수정	윤상건
	GameRunner::getStatus();	현재 진행중인 게임의 StatusManager 객체를 반환	윤상건
include/Snake- Game/PlayerGameManager.h src/PlayerGameManager.cpp	WINDOW* windowScoreBoard	스코어 보드 UI 윈도우를 담고있는 포인터 변수	구형모
	WINDOW* windowMissionBoard	미션 보드 UI 윈도우를 담고있는 포인터 변수	구형모
	WINDOW* windowPause	일시정지 UI 윈도우를 담고있는 포인터 변수	구형모
	void initializeScoreBoard();	스코어 UI 윈도우를 초기화 하는 함수	구형모
	void initializeGoalBoard();	미션 UI 윈도우를 초기화 하는 함수	구형모
	void updateScoreStatus(const Score& score);	스코어 UI 윈도우에 스코어를 반영 및 그려주는 함수	구형모



	void updateMissionStatus(cosnt Mission& mission);	미션 UI 윈도우에 미션 상황 반영 및 그려주는 함수	구형모
	void showPauseWindow();	일시정지 UI 윈도우를 초기화 하는 동시에 UI 를 그려주는 함수	구형모

6단계 구현 내용

맵 에디터 구현을 위한 MapEditor 클래스를 생성하였다. MapItem 구조체를 공유하여 MapManager 클래스를 통해 Map을 편집하고, FileManger를 통해 최종 저장한다.

6단계 목표를 달성하기 위해 수정된 코드 내용은 다음과 같다.

수정한 파일 명	작성된 클래스/함수	간략 설명	개발자
include/Primitives/ MapEditor.h src/MapEditor.cpp	class MapEditor;	맵 에디터 기능추가를 위한 클래스 하나의 MapItem을 공유	김태윤
	void edit();	맵 편집의 핵심기능을 담당, 사용자의 키보드 입력에 따른 맵 편집 기능을 제공	김태윤
	void showInputWindow();	최초 사용자가 맵의 너비와 높이, 이름을 입력하는 Window를 생성	김태윤
	void initializeWindow();	Window 표시전 초기화	김태윤
	void validateWindow();	현재 터미널 창 크기를 검증하여 window 생성 가능 여부를 파악	김태윤
	void initDrawMap();	Map 요소에서 Wall과 Empty를 구분하여 Ncurses로 맵 시각화	김태윤
	void initSetMap();	사용자에게 입력받은 WH에 맞는 Map 테두리 wall을 생성	김태윤
	void initializeColors();	터미널 색상 사용을 위해 초기화	김태윤
	void autoSetWall()	파일 저장하기 전 사용자가 생성한 Wall이 Wall인지, Immune Wall인지 판별하여 자동으로 지정	김태윤
include/Snake- Game/GlobalStateManager.h src/GlobalStateManager.cpp	class GlobalStateManager;	메뉴화면, 게임 시작 이나 맵 에디터, 커스텀 게임 시작 등 다양한 기능들을 시작 할 수 있는 클래스	구형모
	WINDOW* windowIntro	게임 시작전 맵 정보를 보여주는 인트로 화면을 담는 포인터 변수	구형모



	void run();	main함수에서 이 함수를 호출하여 모든 로직 동작. 메뉴 화면을 보여줌	구형모
	void startGame();	메뉴에서 "Game Start"를 선택하면 불러지는 함수. 4개의 맵을 순차적으로 플레이 하도록 구현함.	구형모
	void startGame(std::string mapDir, std::string username);	주어진 맵으로 게임을 시작 하도록 구현 되어있는 함수. 매 게임 시작전에는 showMapIntro() 함수를 호출함.	구형모
	void startMapIntro(std::string mapDir, std::string username);	게임 시작전 맵 정보를 사용자에게 보여주는 함수	구형모
	void startgameWithAi();	메뉴에서 "P vs AI"를 선택 하면 불러지는 함수. AI와 게임을 하는 모드로 진입.	구형모
	void startMapEditor();	메뉴에서 "Edit Map" 을 선택하면 불러지는 함수. 맵 에디터 시작함	구형모
include/Snake-Game/MenuManager.h src/MenuManager.cpp	class MenuManager;	메뉴 화면을 그리고 그에 따른 로직을 처리하는 클래스	구형모
	void showMenu();	메뉴를 보여주는 함수	구형모
	void destroyMenu();	메뉴를 없애주는 함수	구형모
	int maxHeight;	화면 높이 저장하는 변수	구형모
	int maxWidth;	화면 너비 저장하는 변수	구형모
	std::vector<std::string> modesName;	메뉴 화면에 표시할 아이템의 제목들 담는 객체	구형모
	WINDOW* menuWindow;	윈도우 화면 UI를 담는 객체	구형모
	void initializeMenu();	메뉴 화면을 초기화 하는 함수	구형모
	void printMenu(int needTobeHighlighted);	메뉴 화면 내 아이템들을 그려주는 로직 및 방향키로 탐색하는 로직이 담겨있는 함수.	구형모
	void startMode(int mode);	메뉴를 선택하면 불리는 함수. 아이템에 따라 알맞은 기능들이 실행됨.	구형모



7단계 구현 내용

GA 와 MLP 를 구현하여 학습을 진행했다. 자세한 내용은 앞서 설명했으므로 생략한다.

7단계 목표를 달성하기 위해 수정된 코드 내용은 다음과 같다. 이때, 해당 구현 내용을 자세히 설명하는 것은 프로젝트의 주 목표에 비해 너무 복잡하고 거리가 있으므로 각각의 메소드들에 대한 자세한 설명은 생략했다.

수정한 파일 명	작성된 클래스/함수	간략 설명	개발자
include/GA-MLP/Primitives/Sigmoid.h	struct Sigmoid;	활성화 함수 중 하나인 Sigmoid함수 functor를 가지고 있는 클래스	윤상건
	struct dSigmoid;	활성화 함수 중 하나인 Sigmoid의 도함수 functor를 가지고 있는 클래스	윤상건
include/GA-MLP/GA.h src/GA.cpp	class GA;	Genetic Algorithm을 구현한 클래스	윤상건
include/GA-MLP/MLP.h src/MLP.cpp	class Perceptron;	MLP에 사용될 각 Perceptron을 구현한 클래스	윤상건
	class MLP;	Multilayer Perceptron을 구현한 클래스	윤상건
include/GA-MLP/Snake-AI-Learner.h src/Snake-AI-Learner.cpp	class SnakeAILearner;	GA클래스와 MLP클래스 그리고 GameRunner클래스를 이용하여 Snake Game학습을 진행하는 클래스	윤상건
include/GA-MLP/Snake-AI.h src/Snake-AI.cpp	class SnakeAI;	GameRunner를 상속받은 인공지능 게임 진행 클래스	윤상건
include/Snake-Game/AiGameManager.h src/AiGameManager.cpp	class AiGameManager;	기능은 PlayerGameManager랑 동일 하나 스코어보드, 메뉴 화면이 삭제되고 인공지능 게임 화면이 오른쪽에 추가됨	구형모

2.2.3 활용/개발된 기술


- ncurses
 - 사용자 화면에 UI를 그려주기 위해 사용
- STL
 - 데이터를 효율적으로 관리하기 위한 컨테이너로써 활용
 - std::vector, std::set, std::deque, std::queue를 주로 활용했다.



- 효율적인 정렬을 위해 사용
 - algorithm헤더에서 제공하는 std::sort()를 이용하여 효율적인 정렬 알고리즘을 이용할 수 있었다.
- fstream
 - txt기반 맵/유저 불러오기, 저장을 구현하기 위한 파일 입출력 용도로 활용
 - ifstream, ofstream을 주로 활용하였다.
- 새롭게 고안한 알고리즘
 - Snake의 이동 알고리즘
 - Snake의 이동을 구현하기 위해 deque자료구조를 활용한 알고리즘을 구현했다.
 - Snake 게임 학습
 - Snake Game을 플레이 하는 인공지능망을 학습시키기 위해 Genetic Algorithm 과 MLP를 활용한 학습 알고리즘을 구현했다.

2.2.4 현실적 제한 요소 및 그 해결 방안

- 인공지능 학습에서의 한계
 - 사용자와 인공지능 간의 대전을 수행하는 기능을 구현하기 위해 Mission을 수행하는 인공지능 구현을 목적으로 했다. 하지만 Mission을 달성하도록 하기 위한 인공지능 학습을 하기 위해서는 강화학습과 같은 기술의 적용이 필요했는데, 시간적 부족으로 인해 Mission달성을 하기 위한 인공지능 학습은 구현하지 못했다. 이를 해결하기 위해 비교적 쉽게 구현할 수 있는 GA와 MLP를 이용해 학습하는 것으로 학습 방법을 실현 가능한 방법으로 수정했고, 이들을 통해 Mission을 달성하는 방식이 아닌 특정 점수를 최대화 하는 것으로 기능을 구현할 수 있었다.
- Ncurses 라이브러리와 터미널의 한계
 - 터미널의 창 크기를 Ncurses가 조절할 수 없어서 터미널이 요구하는 크기보다 작다면 사용자가 일일이 늘려서 다시 실행해야 하는 단점이 있음.
- 기본 Filesystem 모듈의 부재로 인한 한계
 - 현재 구현에 사용된 c++14에는 OS 파일 시스템에 접근할 수 있는 기본 모듈(라이브러리)가 없어 직접 구현하여야 하는데 구현 시에 OS 종속적인 부분이 있어, 프로그램의 크로스 플랫폼 지원을 훼손시키는 문제점이 있기에 예외 처리를 하여, 각 OS의 파일 시스템에 맞게 파일 디렉토리 목록을 출력하는 코드를 별도로 구현하였다.

 국민대학교 소프트웨어학부 C++ 기말 프로젝트	결과보고서		
	프로젝트 명	Snake Game구현 및 맵 제작, 인공지능 대전 기능 구현	
	팀 명	학습한 구렁이	
	Confidential Restricted	Version 1.4	2020-06-20

2.2.5 결과물 목록

- include/GA-MLP/Primitives/Sigmoid.h
 - 인공지능 구현에 사용된 활성화함수인 Sigmoid 함수와 그 도함수 functor 를 가진 클래스를 구현하기 위한 헤더 파일
- include/GA-MLP/GA.h
 - 인공지능 구현에 사용된 Genetic Algorithm 을 구현한 클래스 GA 가 정의되어 있는 헤더 파일
- include/GA-MLP/MLP.h
 - 인공지능 구현에 사용된 Multilayer Perceptron 을 구현한 클래스 MLP 가 정의되어 있는 헤더 파일
- include/GA-MLP/Snake-AI-Learner.h
 - 인공지능 학습을 위해 구현한 클래스 SnakeAILearner 가 정의되어 있는 헤더 파일
- include/GA-MLP/Snake-AI.h
 - 인공지능의 Snake Game 플레이를 구현한 클래스 SnakeAI 가 정의되어 있는 헤더 파일
- include/Snake-Game/Primitives/Item.h
 - Snake Game 의 맵 각 좌표에 존재하는 각 종류들이 enum 타입으로 구현되어 있는 헤더 파일
- include/Snake-Game/Primitives/MapItem.h
 - 게임 맵에 관련된 모든 Item(이름, 생성일, 내용, 조건)등을 모두 담아 통합한 구조체가 구현되어 있는 헤더파일
- include/Snake-Game/Primitives/MapManager.h
 - 게임 맵 데이터를 2 차원 벡터형태로 가공하여 저장하는 기능이 구현되어 있는 헤더파일
- include/Snake-Game/Primitives/Mission.h
 - 한 게임의 미션 달성 여부를 저장하는 Mission 구조체가 구현되어 있는 헤더 파일
- include/Snake-Game/Primitives/Point.h
 - 2 차원 좌표 관리 및 조작을 더 쉽게 할 수 있도록 하기 위한 2 차원 좌표 구조체 Point 가 구현되어 있는 헤더 파일
- include/Snake-Game/Primitives/Score.h
 - 한 게임의 스코어를 저장하는 Score 구조체가 구현되어 있는 헤더 파일
- include/Snake-Game/Primitives/UserItem.h



- 사용자 정보를 저장하는 UserItem 구조체가 구현되어 있는 헤더 파일
- include/Snake-Game/Snake-Game.h
 - Snake 를 관리하는 Snake 클래스와 Snake Game 을 관리하는 GameRunner 클래스가 정의되어 있는 헤더 파일
- include/Snake-Game/StatusManager.h
 - 한 게임의 미션 달성 여부와 스코어를 관리하는 StatusManager 클래스가 정의되어 있는 헤더 파일
- include/Snake-Game/FileManager.h
 - 맵과 유저정보를 불러오고 저장하는 FileManager 클래스가 정의되어 있는 헤더 파일
- include/Snake-Game/MapEditor.h
 - 사용자 맞춤 맵을 생성해 편집할 수 있는 MapEditor 클래스가 정의되어 있는 헤더파일
- include/Snake-Game/GlobalStateManager.h
 - 기능을 실행 하거나 메뉴화면을 호출 및 종료하는 등 프로그램 흐름을 전반적으로 제어하는 기능이 정의되어 있는 헤더파일
- include/Snake-Game/PlayerGameManager.h
 - Ncurses 를 활용하여 플레이어가 게임을 할 수 있도록 맵을 그려주거나 입력을 받는 기능이 정의되어 있는 헤더파일
- include/Snake-Game/AiGameManager.h
 - Ncurses 를 활용하여 플레이어와 인공지능이 게임을 할 수 있도록 맵을 그려주거나 입력을 받는 기능이 정의 되어 있는 헤더파일
- include/Snake-Game/MenuManager.h
 - Ncurses 를 활용하여 사용자가 원하는 기능을 실행하도록 구현한 클래스가 정의 되어 있는 헤더파일
- src/GA.cpp
 - 인공지능 구현에 사용된 Genetic Algorithm 을 구현한 클래스 GA 가 구현되어 있는 C++파일
- src/MLP.cpp
 - 인공지능 구현에 사용된 Multilayer Perceptron 을 구현한 클래스 MLP 가 구현되어 있는 C++파일
- src/Snake-AI-Learner.cpp
 - 인공지능 학습을 위해 구현한 클래스 SnakeAILearner 가 구현되어 있는 C++파일



- src/Snake-AI.cpp
 - 인공지능의 Snake Game 플레이를 구현한 클래스 SnakeAI 가 구현되어 있는 C++ 파일
- src/Snake-Game.cpp
 - Snake 를 관리하는 Snake 클래스와 Snake Game 을 관리하는 GameRunner 클래스가 구현되어 있는 C++파일
- src/StatusManager.cpp
 - 한 게임의 미션 달성 여부와 스코어를 관리하는 StatusManager 클래스가 구현되어 있는 C++파일
- src/FileManager.cpp
 - 맵과 유저정보를 불러오고 저장하는 FileManager 클래스가 구현되어 있는 C++파일
- src/MapEditor.cpp
 - 사용자 맞춤 맵을 생성해 편집할 수 있는 MapEditor 클래스가 구현되어 있는 C++파일
- src/GlobalStateManager.cpp
 - 기능을 실행 하거나 메뉴화면을 호출 및 종료하는 등 프로그램 흐름을 전반적으로 제어하는 기능이 구현 되어 있는 C++ 파일
- src/PlayerGameManager.cpp
 - Ncurses 를 활용하여 플레이어가 게임을 할 수 있도록 맵을 그려주거나 입력을 받는 기능이 구현 되어 있는 C++ 파일
- src/AiGameManager.cpp
 - Ncurses 를 활용하여 플레이어와 인공지능이 게임을 할 수 있도록 맵을 그려주거나 입력을 받는 기능이 구현 되어 있는 C++ 파일
- src/MenuManager.cpp
 - Ncurses 를 활용하여 사용자가 원하는 기능을 실행하도록 만든 클래스가 구현 되어 있는 C++ 파일



프로젝트 명	Snake Game구현 및 맵 제작, 인공지능 대전 기능 구현	
팀 명	학습한 구렁이	
Confidential Restricted	Version 1.4	2020-06-20

3 자기평가

• 윤상건

이번 C++언어로 Snake Game 을 개발하는 팀 프로젝트에서 팀장을 맡아 프로젝트를 시작할 때 프로그램 설계, 버전 관리, 코딩 규칙, 커밋 규칙 등을 정하고 각 팀원들과 조율을 통해 역할 분담을 진행했다. 개발자로서의 역할로는 Snake Game 의 게임 로직 부분을 담당했다. 즉, Snake Game 을 한 프레임씩 사용자의 명령을 받아 동작시키고 사용자에게 출력해야 되는 정보를 규칙에 맞게 반환하는 클래스인 GameRunner 를 주로 개발했다. 이후에 부가기능 구현에서 게임 인공지능 개발을 진행했다. 평소 C++언어를 이용하여 알고리즘 문제를 해결하는 것에 관심이 많았고 경험도 많았기 때문에 게임 로직을 구현해 내는 것에는 큰 어려움이 없었지만, C++을 이용하여 프로젝트를 진행해 본 적은 없었기 때문에 프로젝트의 구조, 테스트 코드 작성 방법 등에 대해 약간의 어려움이 있었다. 하지만 여러 인터넷 자료를 찾아가며 C++을 이용한 프로젝트의 디렉토리 구조는 어떻게 구성하는지, 많이 쓰이는 Makefile 의 형식은 어떤지, 테스트 코드를 어떤 방식으로 작성하는지 등 많은 것을 배워 나갈 수 있었다. 특히, Makefile 은 처음 활용해 봤는데, 다양한 방식으로 매크로 기능을 활용할 수 있어서 흥미로웠다. 또한 다양한 OS 환경에도 작동할 수 있게 Makefile 을 설계하는 방법도 배울 수 있었다. 아쉬웠던 점으로는 인공지능을 학습시킬 때에도 어려움을 겪었다는 것이다. 주어진 Snake Game 에는 Growth아이템 뿐만 아니라 Poison 아이템, Gate 등 다양한 요소가 존재해서 학습이 어려웠다. 결국 크게 만족스럽지 못한 학습 결과가 나왔지만, 좀 더 시간이 있었다면 Genetic Algorithm 에서의 교배 규칙 개선, 신경망 개선, 높은 수준의 컴퓨팅 파워로 학습 진행 등의 방법으로 크게 만족할 만한 결과를 낼 수 있으리라 생각한다. 시간적 문제를 제외하고, 프로젝트 운영은 잘 진행되었던 것 같고 스스로 또한 맡은 부분을 모두 빠르게 달성했기 때문에 전반적으로 만족스럽게 프로젝트를 완료한 것 같다.

• 구형모

C++ 프로그래밍을 통해 배웠던 개념들을 알맞게 활용하는 법을 배워보는 기회였던 동시에 Ncurses 를 활용한 터미널 UI 를 직접 구현해보는 기회가 된 프로젝트였던 것 같다. 본인은 GUI 로직을 구현하는 역할을 맡았다. 안드로이드 개발할 당시에는 매우 편하게 GUI 를 Java 나 xml 로 작성하였지만, 본 프로젝트에서 c++과 ncurses 라이브러리를 활용해서 작성하려 하니 가히 지옥이 다름없었다. 하지만 시간이 지날수록 뿌듯함은 점점 더 커져만 갔다. 예를 들어 게임 화면 하나 작성하는 데에 하루 종일 걸렸다면 그 뒤 명세가 바뀌어 레이아웃을 다시 구성해야 할 때는 몇 시간, 그 후 메뉴 화면을 만들 때에는 한 시간 정도 만에 똑딱 만들게 되었다. 많은 삽질(잘 안 풀려서 이것 저것 시도해보는 행위)을 통해 얻은 지식과 실력이 늘어감에 따라 불행을 행복으로 바꾸어 나가는 나 자신을 보고 뿌듯하기도 하였다. 프로젝트 형상 관리는 Git 을 이용해 관리하였으며, Conflict 를 피하고자 브랜치를 나누어 master 브랜치에 merge 하는 방식으로 진행했다. 가장 좋았다고 생각했던 점은 Business Layer(게임 로직) 와 UI Layer(Ncurses 로 터미널에 표현) 가 명확하게 나뉘어 있어 각 역할을 맡은 사람들이 원활하게 작업을 할 수 있다는 점이 좋았다. 추후 여러 개발자가 공존하는 큰 기업에서도 이러한 경험들을 생각하며 적용할



예정이다. 프로젝트의 아쉬운 점을 굳이 꼽자면 라이브러리가 한정적인 점이라고 생각한다. Ncurses 말고도 Qt, WinApi 등등 다양한 라이브러리를 활용할 수 있었다면 좀 더 멋진 게임이 탄생하지 않았을까 하는 욕심이 프로젝트를 진행하는 중간중간 들기도 하였다. 결론적으로는 수업에 배운 내용을 전반적으로 적용하고 알맞게 사용해보면 C++ 언어의 고급적인 스킬들을 키워볼 수 있었던 유익한 프로젝트임은 틀림없다고 생각한다.

• 김태윤

이번 C++ 프로젝트에서 맵과 유저 데이터를 사용하기 위한 파일 입출력 기능과 Ncurses 를 사용하여 사용자 정의 맵을 생성하고 편집하는 기능을 개발하는 역할을 맡아 이를 구현하였다. OS 파일 시스템이나 GUI 는 파이썬으로 다루어본 경험 밖에 없는데, 파이썬 대비 빈약한 기본 라이브러리 환경과 터미널 기반 프로그램의 제약으로 기본적인 초기 기능 구현과정에서 정말 많은 시행 착오를 겪으면서 개발을 진행하였다. 이러한 어려움 속에서 프로젝트 초기 명확한 역할 분담으로 큰 혼란없이 원활한 팀 프로젝트를 진행할 수 있었고, 프로젝트 전반의 git 활용으로 branch, merge, pull request 등 협업에 필수적인 개념과 기능을 직접 사용해보면서 익혀 처음에는 미숙 했지만 잘 다룰 수 있게 되어 효율적으로 코드를 관리할 수 있었다. 또한 다른 개발자가 개발한 코드와 내가 개발한 코드를 연동하는 과정에서 모르는 점은 웹에서 찾아보거나 직접 물어보며 실력과 개발자로서의 협업 능력을 키울 수 있던 기회여서 좋았다. 또한 다른 팀원들에게 작성한 코드의 코드 리뷰를 통해 코드와 로직을 개선, 구현하는 데에 있어서 많은 도움을 받아 C++ 프로그래밍 실력을 키울 수 있는 좋은 경험이었다. ncurses 를 조금 더 깊게 분석하여 디자인적인 요소를 개선하거나, Animation 등 동적인 요소를 구현 하였더라면 보다 완성도 높은 재미있는 게임을 만들 수 있겠다 라는 아쉬움이 들기도 하고, 앞으로는 더 실력을 키워 이와 같은 프로젝트에서 메인로직이나 AI 구현 부분에 주도적인 역할을 해야겠다는 생각이 들기도 하였다. 이번 프로젝트는 초기 목표를 모두 달성하여 구현 하였기에 의미있고 좋았던 프로젝트라고 생각된다.



4 참고 문헌

번호	종류	제목 및 출처	발행년도	저자
1	책	C 언어 본색	2011-01-05	박정민
2	PPT	C++ 프로그래밍 주차별 교안 PPT ecampus	미상	임은진
3	웹사이트	NCURSES-Programming-HOWTO http://wiki.kldp.org/wiki.php/NCURSES-Programming-HOWTO	2001-05-22	Pradeep Padala
3	웹사이트	curses programming sample http://egloos.zum.com/clccclcc/v/9527073	미상	미상
4	웹사이트	Snake with Curses in C https://codereview.stackexchange.com/questions/159222/snake-with-ncurses-in-c	2017-04	mnoronha



프로젝트 명	Snake Game구현 및 맵 제작, 인공지능 대전 기능 구현	
팀 명	학습한 구렁이	
Confidential Restricted	Version 1.4	2020-06-20

5 부록

5.1 사용자 매뉴얼

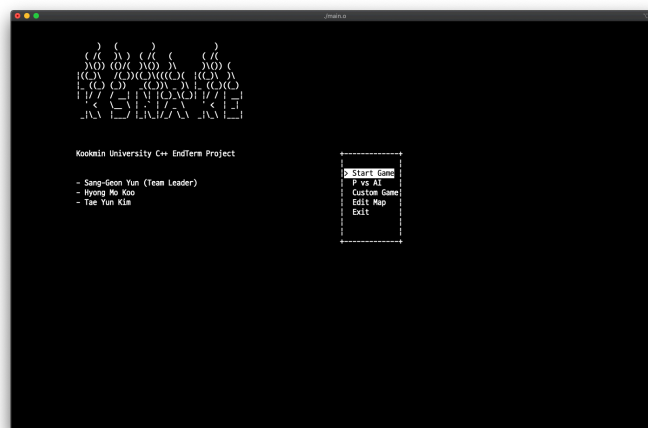
```
taeyun@TaeYunui-MacBook-Pro: ~/github-kmu/Snake-Game/bin
taeyun ~$ cd github-kmu/Snake-Game
taeyun ~/github-kmu/Snake-Game $ master cd bin
taeyun ~/github-kmu/Snake-Game/bin $ master ./main.o
Window size should be bigger than 150 X 30
x taeyun ~/github-kmu/Snake-Game/bin $ master
```

아래의 5.2 의 설치 방법을 참고하여 라이브러리를 설치하고 컴파일 한 뒤 ./main.o 를 실행합니다.

* 터미널 윈도우 크기가 작은 경우 위와 같이 실행이 되지 않습니다.

[전체화면 플레이를 권장합니다.]

0. 메뉴화면



정상적으로 실행이 되면 메뉴화면이 표시되며 각 항목은 방향키로 이동할 수 있습니다.

Level1~4 까지 순차적으로 플레이하고 싶으면 "Start Game"을,

사람과 AI 대전을 하고 싶으면 "P vs AI"를,

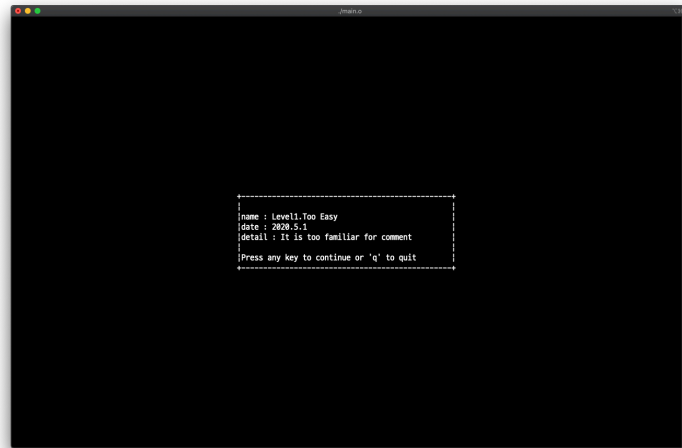
직접 맵을 생성하고 싶으면 "Edit Map"을,

생성한 맵으로 플레이 하고 싶으면 "Custom Game"을

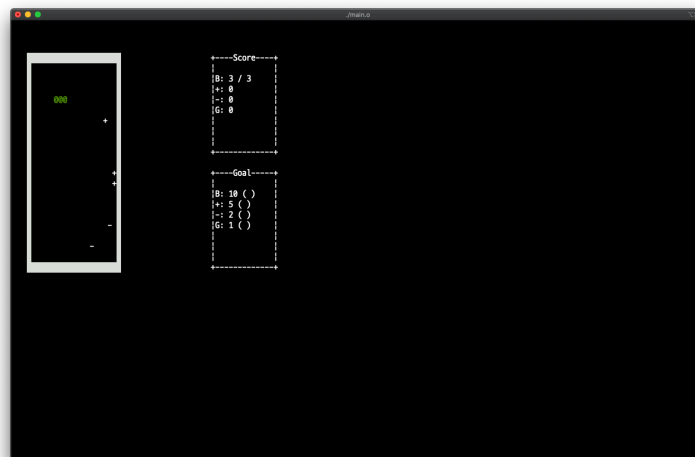
종료하고 싶다면 "Exit"에 초점을 맞춘 뒤

엔터키를 누르면 선택한 항목으로 이동합니다.

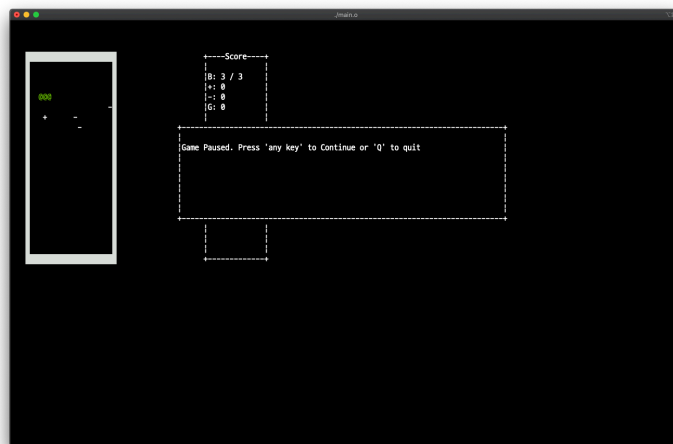
1. Start Game



“Start Game”을 눌러 게임을 시작하면 맵 정보가 나오며, q 키를 제외한 키보드의 아무키를 누르면 게임이 시작됩니다. 만약 이전화면으로 돌아가고 싶다면 q 키를 눌러 돌아갈 수 있습니다.



게임이 시작되면 방향키로 Snake 를 자유롭게 움직일 수 있으며, 우측 사이드 바에 점수와 미션이 표시됩니다.





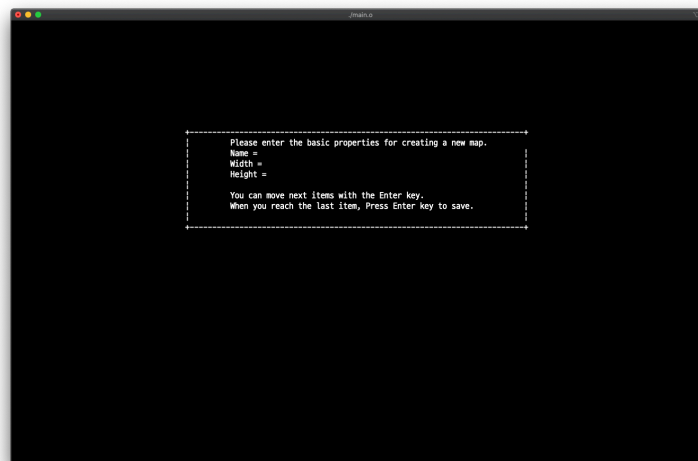
게임을 일시 정지하고 싶은 경우 q 키를 눌러 일시정지 할 수 있으며,
q 키를 다시 누르면 일시정지가 해제 됩니다.

2. P vs AI

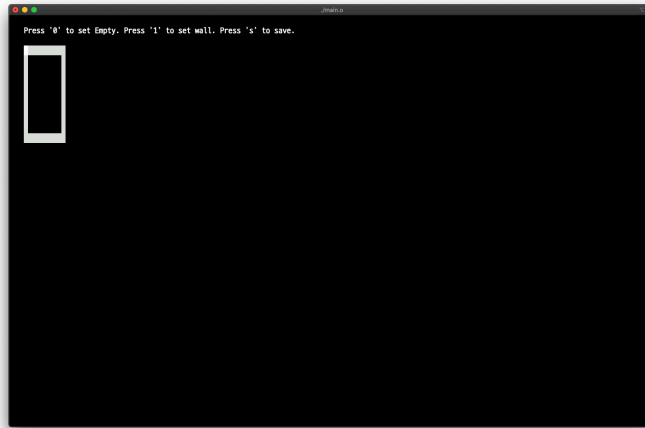


인공지능과 대결할 수 있는 기능입니다.
조작방법은 위 게임과 동일하며, 좌측에는 사용자가 플레이하고 있는 게임,
우측에는 인공지능으로 플레이하고 있는 게임이 표시됩니다.
사용자가 인공지능보다 오래 살아남게 되면 승리하게 됩니다.

3. Edit Map



사용자 정의 맵을 생성하기전 기본정보 Name, Width, Height 를 입력합니다.
다음 항목으로 이동은 엔터 키로 할 수 있으며 마지막 항목에서 엔터 키를 입력할 경우, 유효성
검증을 통과하면 자동으로 맵 편집기 화면으로 이동합니다.



방향키로 이동을 할 수 있으며,

0 번 키를 누르면 빈공간으로, 1 번 키를 누르면 벽으로 설정할 수 있습니다.

맵 편집을 마치면 s 키를 눌러 저장하면 됩니다.

저장이 완료되면 프로그램이 종료되고, 다시 프로그램을 실행하여 "Custom Game"항목을 선택하면
생성한 맵으로 플레이할 수 있습니다.

5.2 설치 방법

1. 라이브러리 설치

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

2. 컴파일

```
Cd [PROJECT DIRECTORY]  
make
```

3. 실행

```
cd bin  
./main.o
```