**C++프로그래밍**

**프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *NcurSnake* |
| 팀 명 | *02분반 17조* |
| 문서 제목 | 최종 결과보고서 |

|  |  |
| --- | --- |
| **Version** |  |
| **Date** | 17 |

|  |  |
| --- | --- |
| **팀원** | 신현우  김나현  장원재 |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 C++프로그래밍 수강 학생 중 프로젝트 “Ncurses-Snake”를 수행하는 팀 “17조”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “17조”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 최종보고서-NcurSnake.doc |
| **원안작성자** | 신현우, 김나현 |
| **수정작업자** | 신현우, 김나현 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2023-06-14 | 장원재 |  |  | 아이디어 제시 |
| 2023-06-14 | 신현우 | 1.1 | 최초 작성 | 과제 마일스톤 및 개발 목표 작성 |
| 2023-06-15 | 신현우 | 1.1 | 내용 수정 | 개발 목표추가 |
| 2023-06-16 | 신현우 | 1.2 | 내용 수정 | 개발 내용 및, 구현 내용 추가 |
| 2023-06-17 | 김나현 | 1.3 | 내용 수정 | 아이템 및 스테이지 개발 내용 추가 |
| 2023-06-17 | 김나현 | 1.4 | 내용 수정 | 렌더링 관련 활용 기술, 개발 내용 추가 |
| 2023-06-18 | 신현우 | 1.5 | 최종 작성 | 시스템 구조 및 설계 작성  활용된 기술 작성  제한 요소 및 해결방안 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

## 소개

C++와 Ncurses 라이브러리를 활용하여 만든 게임입니다. 화면상의 표현되는 Snake는 플레이어가 방향키를 이용해 조작할 수 있습니다. 플레이어는 주어진 시간 내에 어떤 동선으로 움직여야 효율적으로 아이템을 먹을 수 있을지 생각하며 미션을 수행해 나가는 게임입니다.

플레이어는 화면상 각각 다르게 표시되는 아이템을 먹어 미션을 클리어 하기 위해 여러 장애물을 피해 움직여야 합니다.

# 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 |

### 개요

* 컴포넌트 디자인 패턴을 통해 게임을 구현한다.
* 각각의 기능을 가진 컴포넌트는 GamaManager라는 Hub Class에서 조립하며 사용한다.
* 예를 들어, snake는 움직이거나, 몸을 길게 하거나, 꼬리를 자르는 일만 수행해야 하고,
* Map은 snake가 어떻게 구현되어 있는지 상관없이, 맵에 특정 좌표에 있는 데이터를 반환하거나 설정하고, 화면에 출력할 수 있어야 한다.
* 전체적인 게임 루프와 디자인은 유니티 엔진의 아키텍처를 참고한다.

### 1단계 map의 구현 목표

* 사용자의 터미널 WINDOW 크기에 맞는 2차원 배열을 동적으로 할당한다.
* Map의 있는 data와, ncurse를 통해 데이터를 출력할 Board Class를 만든다.
* 2차원 배열은 동적 메모리 할당을 통해 초기화 함으로써, 런타임에서도 스테이지 변화에 따른 맵의 크기 변경이 가능할 수 있도록 한다.
* 정적으로 생성되어 이벤트가 없을시 업데이트를 할 필요가 없는 데이터들(etc. item, gate)은 map의 2차원 배열을 통해 데이터를 입력 혹은 출력한다.

### 2단계 snake 표현 및 조작 구현 목표

* Snake는 연결된 표현법을 통해 구현한다.
* 스네이크의 정보는 map에 따로 저장하지 않는다. 단, snake가 이동할때마다 맵의 정보를 읽으며 필요한 처리를 한다.
* Snake의 head부분과 body부분을 구분은 컬러를 다르게 하는것으로 구현한다.
* Snake는 움직이고, 꼬리를 자르거나, 머리를 추가하는 역할만 한다.
* Snake는 자신의 머리가 몸 안에 있다면 죽어야 한다.
* Snake는 Score Board를 위한 본인의 size와 head의 coordinate를 저장하고 있어야 한다.

### 3, 4단계 Item과 Gate의 구현 목표

* Item, Gate의 생성과 소멸 주기는 Spawner라는 Class를 통해 관리
* Spawner 클래스는 Item의 정보를 업데이트 하고, 요청에 따라 오래된 아이템은 삭제하고 새로운 아이템은 생성하는 역할을 한다.
* Item의 종류와 정보는 GameManager를 통해 Snake와 Map으로 전달된다.
* 전달 시 GameManager에서 해당 아이템 타입에 맞춰 적절한 코드를 실행하게 한다.
* 아이템이나 게이트 생성시, Spawner 클래스는 랜덤하게 생성된 아이템의 좌표를 던져주고 그것을 받은 GamaManager에서 Map의 좌표와 비교해 유효하지 않은 좌표라면 다른 좌표에 생성할 것을 요청한다.
* Gate의 기능 구현도 Spawner를 통해서 진행한다.
* Spawner는 게이트 생성시 출구의 위치를 설정해야 한다. 만약 벽이 가장자리가 아닐 경우엔 Snake가 입구에서 들어오는 방향에 따라 출구의 목적지와 방향이 정해져야 한다.
* 가장자리가 아닐 경우, 목적지는 Snake의 입구에서 들어오는 방향으로부터, 출구의 목적지가 벽이 아닐동안, 반시계방향으로 이동시킨다. 이 때 스네이크가 출구에서 나온 후 이동할 방향은 목적지의 방향과 같아야 한다.

### 5단계 Score 및 Mission Board의 구현 목표

* GameManager로부터, 각각의 component의 정보를 받아와, 화면에 출력한다.
* 각 아이템 조건에 따라 현재 점수를 계산하고, 미션에 맞는 stage로 넘어갈 수 있게 하는 함수를 구현해야 한다.

## 개발 내용 및 결과물

### 개발 내용

|  |  |  |
| --- | --- | --- |
| 클래스 이름 | 개발자 | 개발 내용 |
| Game Manager | 신현우 | 1. 생성자 호출시 메인 게임을 위한 초기화기능 추가 2. 각각의 컴포넌트들을 초기화하는 기능 추가 3. 메인 게임 루프를 위한 업데이트 함수 기능 추가 4. 컴포넌트 변수를 저장하고 호출해 쓰기 위한 함수 추가 5. 게임 상태가 유효한지 검사하는 함수 추가 |
| Board | 신현우 | 1. 2차원 배열(map)의 메모리를 WINDOW 크기에 맞춰 동적할당 하는 기능 추가 2. 출력을 위한 window 초기화, 업데이트 하는 기능 추가 3. Map에 저장된 데이터를 바꾸거나 읽을 수 있게 하는 함수 4. Map의 있는 정보를 업데이트 해 출력하는 기능 추가 |
| Board | 김나현 | 1. Stage에 출력될 맵 구현 추가 2. Attribute를 이용한 아이템 및 게이트의 렌더링기능 추가 |
| Snake | 신현우 | 1. 연결된 표현법을 사용하기 위해 Node 기능을 할 Body클래스 추가 2. 스네이크의 길이를 추가하거나, 삭제하는 기능 추가 3. 스네이크의 머리를 특정 위치로 이동시키게 하는 기능 추가 4. 스네이크의 몸통이 머리를 따라가게 하는 기능 추가 5. 스네이크 유효성 검사하는 함수 추가 |
| Misc | 신현우 | 1. 게임의 여러가지 초기화 세팅을 한 곳에서 할 수 있도록 함 |
| Spawner | 신현우 | 1. 아이템을 생성시키고, 해당 아이템의 정보를 tracking하는 기능 추가 2. Item의 정보를 저장하기 위한 구조체 추가 3. Gate의 저장된 정보를 읽어오게 하기 위한 기능 추가 |
| Stage | 김나현 | 1. 반복문을 통해 사용자 윈도우 크기에 맞게 할당된 배열의 사이즈 안에서 생성될 맵 단계별 설계 2. Static 함수를 통해 별도의 객체 생성 없이 생성된 맵의 정보를 수정하는 기능 추가 |
| ScoreBoard | 신현우 | 1. 화면에 출력되고, 갱신해야 할 정보를 저장하는 멤버필드 추가 2. 스코어 정보와 미션 정보 업데이트 하고 출력하는 기능 추가 |

### 시스템 구조 및 설계도

### 

**GameManager 클래스**

**-개발자 : 신현우**

**-소스 파일: GameManager.hpp**

**-주요 기능: Component들의 정보를 수집, 처리하는 역할을 합니다.**

**-method**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **리턴 타입** | **식별자** | **설명** |
| **private** | **void** | **initSnakeGame** | **게임초기화 및 그래픽 요소 초기화** |
| **private** | **void** | **initMisc** | **런타임에 결정되는 설정값 관련 요소 초기화** |
| **private** | **void** | **moveSnake** | **INPUT을 받아 스네이크를 이동시킴`** |
| **private** | **void** | **validate** | **스네이크 머리의 좌표와 맵의 데이터를 대조해 게임의 유효성을 검사함** |
| **private** | **void** | **applyItemBlock** | **스네이크 머리의 좌표와 맵의 데이터를 대조해 아이템 효과를 적용시킴** |
| **private** | **void** | **spawnItem** | **Spanwer 컴포넌트에게 아이템을 생성 가능한지 물어보고 가능하다면 생성을 지시함.** |
| **private** | **void** | **updateItemInfo** | **Spanwer 컴포넌트가 가지고 있는 각각의 아이템 정보를 업데이트 시키게 함.** |
| **private** | **void** | **updateScoreInfo** | **Score 컴포넌트가 가지고 있는 스코어에 대한 정보를 업데이트 시킴** |
| **private** | **void** | **displayBoard** | **Board 컴포넌트에게 버퍼에 쌓인 내용을 출력하게 함** |
| **private** | **void** | **displayScore** | **Score 컴포넌트에게 버퍼에 쌓인 내용을 출력하게 함** |
| **Public** | **void** | **awake** | **모든 필드를 초기화하는 생성자 기능을 가진 함수를 객체의 소멸 없이 원하는 시점에 호출하게 함** |
| **Public** | **void** | **update** | **Update가 필요한 모든 컴포넌트들에게 순차대로 업데이트를 지시함.** |
| **public** | **void** | **onDisable** | **소멸자 기능을 가진 함수를 원하는 시점에 호출하게 해줌** |

**Board 클래스**

**-개발자 : 신현우, 김나현**

**-소스 파일: Board.hpp**

**-주요 기능: 메인 게임이 진행되는 Map의 데이터와, 해당 데이터가 출력될 Window를 관리합니다.**

**-주요 알고리즘: 2차원 배열 동적 할당을 통해 플레이어의 게임 실행 환경에 따라 유연하게 맵이 생성되게 하고 또한 2차원 배열을 사용함으로써 특정 좌표에 대한 정보를 O(1)시간에 업데이트하게 했습니다.**

**-method**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **리턴 타입** | **식별자** | **설명** |
| **private** | **void** | **printMap** | **Window에 쌓인 버퍼를 출력** |
| **Public** | **void** | **awake** | **모든 필드를 초기화하는 생성자 기능을 가진 함수를 객체의 소멸 없이 원하는 시점에 호출하게 함.** |
| **Public** | **void** | **update** | **전달받았던 모든 정보를 기반으로 자신을 업데이트함.** |
| **Public** | **void** | **onDisable** | **동적할당받은 메모리를 해제하고, 생성한 윈도우를 지움** |
| **Public** | **void** | **clear** | **이전까지 출력되어 있던것들을 모두 지움.** |
| **Public** | **void** | **buildMap** | **Stage 번호를 인자로 받아서, 해당 스테이지의 해당하는 맵으로 데이터를 새로 초기화함.** |
| **Public** | **void** | **setMapData** | **x, y, block정보를 인자로 받아 해당 좌표의 맵 정보를 설정함** |
| **Public** | **void** | **getMapData** | **X,Y 좌표에 있는 맵의 Block을 리턴함** |

**ScoreBoard 클래스**

**-개발자: 신현우**

**-소스 파일: ScoreBoard.hpp**

**-주요 기능: 메인 게임이 진행되는 동안 Score 관련 정보와 Mission 관련 정보를 저장하고 출력합니다.**

**-사용한 외부 라이브러리: <chrono> :: time 정보를 정확히 계산하기 위해 사용했습니다.**

**-method**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **리턴 타입** | **식별자** | **설명** |
| **private** | **void** | **printStage** | **Stage에 맞는 미션 정보를 출력함** |
| **private** | **void** | **calcTotal\_Score** | **현재 저장된 여러 score field 값에 따라서 최종 점수를 계산함** |
| **private** | **void** | **updateMissionWin** | **Mission 부분 정보를 출력할 Window를 업데이트** |
| **private** | **bool** | **isMissionComplete** | **해당 스테이지의 모든 미션이 성공되었는지 확인함** |
| **private** | **void** | **clearScore** | **최종점수를 제외한 모든 점수를 초기화함. 스테이지가 갱신될 때 사용됨** |
| **public** | **void** | **clear** | **Score Board 내 WINDOW들을 초기화시킴** |
| **public** | **int** | **getCurrStage** | **현재 스테이지가 어딘지 반환함** |
| **public** | **void** | **checkMissionCondition** | **각각의 미션들에 대해 condition을 확인하고 업데이트 시킴** |
| **public** | **void** | **set[score\_field\_name]** | **각각의 score들(snake size, growth item count…)의 정보를 업데이트 함. GameManager를 통해 설정됨.** |
| **public** | **void** | **updateTime** | **Chrono 라이브러리를 통해 설정되는 시간을 업데이트시킴.** |

**Spawner 클래스**

**-소스 파일: Spawner.hpp**

**-주요 기능: 아이템과 게이트의 정보를 저장하고, 조건에 맞게 아이템을 처리하거나 생성해 저장합니다.**

**-사용한 외부 라이브러리:**

**<deque> ::아이템의 생성과 소멸을 순차적으로 다루기 위해 STL덱을 사용했습니다.**

**<cstdlib>, <ctime> :: 아이템 생성시 사용할 난수 발생을 위해 사용했습니다.**

**-method**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **리턴 타입** | **식별자** | **설명** |
| **public** | **Void** | **getRandomPosition**  **(int&, int&)** | **난수를 생성해 레퍼런스 변수에 할당해줌.** |
| **public** | **void** | **awake** | **난수 발생시 사용할 난수의 범위를 설정하고, 시드값을 생성함** |
| **public** | **void** | **spawnGrowthItem**  **(int x, int y)** | **X, Y 좌표위치에 Growth 아이템을 생성함.** |
| **public** | **bool** | **getRottenGrowth**  **(int& x, int& y)** | **Score에 저장되어 있는 Growth 아이템 중 유효기간이 지난 게 있다면 true값을 리턴하고, 레퍼런스 변수에 해당 아이템 좌표를 할당함** |
| **public** | **void** | **spawnPoisonItem**  **(int x, int y)** | **X, Y 좌표위치에Poison아이템을 생성함.** |
| **public** | **bool** | **getRottenPoison (int& x, int& y)** | **Score에 저장되어 있는Poison아이템 중 유효기간이 지난 게 있다면 true값을 리턴하고, 레퍼런스 변수에 해당 아이템 좌표를 할당함** |
| **public** | **void** | **spawnGate**  **(x, y, out\_x, out\_y)** | **Score에 저장되어 있는 Gate의 입구와, 출구 좌표를 설정해 생성함.** |
| **public** | **bool** | **getRottenGate**  **(Gate& result)** | **만약 유효하지 않은 게이트가 있다면 true를 리턴하고 레퍼턴스 변수에 해당 게이트를 할당함** |
| **public** | **void** | **setGateActive**  **(int duration)** | **스네이크가 게이트를 통과할경우 게이트를 활성화 시킴. 해당 활성화 시간이 지난 후에 게이트는 자동으로 파괴됨** |
| **public** | **void** | **getGateDestination** | **Gate를 통과시 Snake가 나오게 될 좌표를 레퍼런스 변수를 통해서 반환함** |
| **public** | **bool** | **canSpawn(Block)** | **Block타입으로 넘어오는 아이템이 생성 가능한지 알려줌.** |

**Snake 클래스**

**-개발자: 신현우**

**-소스 파일: Snake.hpp**

**-주요 기능: 플레이어가 조작할 snake의 정보를 조작합니다.**

**-사용한 자료구조 : 연결 표현법(linked list)**

**-Body: 연결리스트의 Node 역할을 하는 struct입니다.**

**-method**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **리턴 타입** | **식별자** | **설명** |
| **public** | **void** | **awake** | **Misc 클래스에서 초기화된 정보를 받아 Snake의 초기 상태를 설정함** |
| **Public** | **void** | **Move**  **(Direction)** | **Direction 정보를 받아서 snake를 이동함** |
| **public** | **void** | **Move**  **(int x, int y)** | **X,Y좌표로 snake를 이동함.** |
| **public** | **bool** | **isAlive** | **Snake의 head가 body 안에 있는게 아닌지 확인함** |
| **public** | **bool** | **isInSnake**  **(int x, int y)** | **X, Y 좌표가 snake와 겹치지 않는지 확인함** |
| **public** | **void** | **insert** | **Snake의 몸 길이를 늘림** |
| **public** | **void** | **remove** | **Snake의 몸 길이를 줄임** |

**Stage 클래스**

**-개발자: 김나현**

**-소스 파일: Stage.hpp**

**-주요 기능: 2차원 배열 Map의 정보를 해당 스테이지 함수에 주어진 반복문에 따라 초기화했습니다.**

**-알고리즘 : static 함수를 통해 따로 객체의 생성 없이 원하는 스테이지의 정보를 설정할 수 있도록 하였습니다.**

**-method**

|  |  |  |  |
| --- | --- | --- | --- |
| **접근제한자** | **리턴 타입** | **식별자** | **설명** |
| **public** | **void** | **stage1,**  **stage2,**  **stage3,**  **stage4** | **주어지는 현재 최대 배열의 높이, 길이 정보를 받아서 어떤 비율에 어느 방식으로 벽이 생성되게 할 것인지 반복문을 통해서 설정함.** |

**Misc 클래스**

**-소스 파일: Misc.hpp**

**-주요 기능: 여러 위치에서 쓰일 다양한 게임 초기 설정 값들을 저장합니다.**

### 활용/개발된 기술

|  |  |
| --- | --- |
| **라이브러리** | **사용한 기능** |
| **chrono** | **Chrono의 steadyClock 기능, TimePoint Class를 이용해 정확한 초단위 시간을 측정했습니다.**  **초기화 단계에서Chrono::steadyClock::now()를 통해 해당 TimePoint 클래스를 저장해놓았습니다.**  **이 후 경과 시간은Chrono::steadyClock::now()를 통해 현재 시간과 저장된 시간의 차이를 비교해 알 수 있습니다.** |
| **deque** | **아이템의 저장을 다루는 자료구조로써 덱을 사용했습니다.**  **가장 먼저 생성된 아이템부터 없어지게 하거나, random access 반복자를 통해 유연하게 아이템 상태를 업데이트 하기 위한 목적으로 사용했습니다.** |
| **Ncurses** | 1. **initscr() : 바탕이 될 기본 stdscr을 초기화 하는 함수** 2. **refresh() : stdscr의 출력 버퍼를 출력해주는 함수** 3. **mvwprintw(): 사용자 정의 Window에 상대적인 위치로 cursor를 이동시킨 뒤, printf 문법을 적용한 출력 을 넣어주는 함수** 4. **box(): window 경계선을 그려주는 함수** 5. **wborder(): 각 edge나 vertex의 char을 설정해 box를 실행하는 함수** 6. **wattron(): 특정 윈도우에서 출력 양식의 attribute를 설정해주는 함수** 7. **wattroff(): 특정 윈도우에서 출력 양식의 attribute설정을 해제하는 함수** 8. **mvwaddch(): 사용자 정의 윈도우의 상대적 위치로 cursor를 이동 시키고 해당 위치에 char을 출력하게 해주는 함수** 9. **getch(): 플레이어의 키보드 입력을 받게 해주는 함수** 10. **nodelay(): 입력이 없을떄도 getch()가 ERR을 반환하는걸 설정해주는 함수** 11. **cbreak(): 플레이어가 Enter키를 누르지 않을때에도 들어오는 입력을 바로 입력버퍼에 넣어주는 설정을 켜주는 함수** 12. **curs\_set(): cursor의 visibla 옵션을 설정함** 13. **start\_color(): 터미널 출력의 색상을 적용시킴. 해당 터미널이 지원하지 않는다면 적용되지 않음.** 14. **Wbkgd(): window의 백그라운드의 특성(색상 등)을 설정함** |
| **cstdlib**  **time.h** | 1. **srand, rand 함수를 통해 난수발생시킬 수 있음** |
| **csignal** | 1. **게임 루프의 tick 주기를 설정하기 위해 사용** 2. **usleep() 을 통해 nanosecond 로 딜레이를 설정할 수 있음.** |

### 현실적 제한 요소 및 그 해결 방안

### 터미널의 폰트 크기조정 제한

### 2.2.5.1 문제

어떤 환경에서도 실행시킬 수 있는 게임을 만들어 보려고 했으나, ncurses가 가지고 있는 현실적인 제한요소들 때문에 해당 부분의 구현이 쉽지 않았습니다.

사용자가 게임을 실행시킬 시 Window를 통해 sub window크기나, 배열 크기 등은 조절할 수 있었지만 폰트 크기는 조정할 수 없었습니다. 그 문제 때문에, window의 크기는 작아져도 폰트의 크기는 그대로여서, Mission Window의 string 출력 결과물이 잘리는 문제가 나타났습니다.

이 부분때문에, 후반부에 mission window를 설계하는 과정에서 여러 방법으로 시도를 해보았지만 특정 크기 이하의 window에서 게임 초기상태가 초기화 될 경우 어떻게 해야 할 지 현실적으로 제한 요소가 있었습니다.

### 2.2.5.2 해결방안

런타임에서 폰트 사이즈 조절을 하는 부분은 찾지 못했고, 특정 크기 이하의 윈도우에서는 실행을 하지 못하게 하는것도 고려해보았으나, 적절하지 못한 것 같아 게임을 실행하는 유저들에게 경고 문구를 알려주는것으로 해결했습니다.

### 2차원 배열 스테이지 설계 문제

### 2.2.6.1 문제

사이즈가 동적으로 결정되다 보니까 컴파일 되어서 플레이어가 실행 되기 전에 정확한 배열이 어떤 모양일지 알 수 없었습니다.

그래서 그냥 처음부터 맵의 값을 지정해 초기화 하는것만큼 스테이지의 벽 위치를 세밀하게 조정할 수 없었습니다.

### 2.2.6.2 해결방안

동적 할당을 할 때, 배열의 상-하 비율 자체는 항상 일정하게 초기화 될 수 있도록 하였습니다.

이렇게 함으로써, 아주 세밀한 표현까진 힘들었지만 대략적으로 어느 위치에 어느 길이만큼 벽이 생성될 것인지 설계하고 적용할 수 있었습니다.

### 결과물 목록

|  |  |
| --- | --- |
| **파일 이름** | **내용** |
| **Makefile** | **Makefile** |
| **Main.cpp** | **메인 함수** |
| **Src/Board.hpp** | **Board 소스코드** |
| **Src/GameManager.hpp** | **GameManager 소스코드** |
| **Src/ScoreBoard.hpp** | **ScoreBoard 소스코드** |
| **Src/Spawner.hpp** | **Spawner 소스코드** |
| **Src/Misc.hpp** | **Misc 소스코드** |
| **Src/Snake.hpp** | **Snake 소스코드** |

# 자기평가

|  |  |
| --- | --- |
| **신현우** | **c++는 많이 써본적이 없기에 프로젝트를 할 수 있을까 걱정도 많이 했지만 다행히 시간 내에 끝낼 수 있었습니다.**  **한 팀원은 개인 사정으로 참여를 할 수 없는 상황에서 끝낼 수 있을까 걱정도 했지만 다행히 수업시간에 배웠던 내용과 제가 소융대 친구 따라 잠시나마 유니티를 써보며 배웠던 지식들을 살짝 결합해서 무사히 프로젝트를 마칠 수 있었던 것 같습니다.**  **프로젝트를 진행하면서 어려웠던 점은 아무래도 좀 오래 된 라이브러리여서 그런지 정보를 찾기 어려웠던 점이였습니다. 예를 들어 저는 폰트의 사이즈, 혹은 컬러의 opacity라던가 구현하고 싶었던 것들이 있었는데 아쉽게도 터미널 설정이나 플레이어 컴퓨터의 환경에 따라 실행시 결과가 달라지는 부분에 대한 정보가 많이 없다 보니 구현하는데 어려움이 있었던 것 같습니다.**  **프로젝트를 진행하면서 개인적으로 느꼈던 점은 프로그램을 코딩하던 중간에 어떤 구조를 바꾸고 싶을 때 잘못 설계하면 처음부터 모든 구조를 다시 짜야 하는 상황이 오는구나 라는걸 느꼈고 해당 부분에서 시간이 가장 오래 걸렸던 것 같습니다. 그리고 상용 게임엔진은 얼마나 대단한것인가 새삼 다시 느끼게 해줬던 프로젝트 였습니다.**  **김나현 팀원과 소통하면서 코드 스타일에 대한 충분한 설명이 없거나 표현 방식을 통일하지 않으면 프로그램 코딩시 잘못 동작되거나 어려움이 생긴다는걸 느꼈고, 해당 부분에 대한 생각을 다시 할 수 있게 해주었던 좋은 시간이 되었던 것 같습니다.**  **수업시간에 배웠던 내용을 충분히 써볼 수 있는 프로젝트였습니다. 단기간에 많은 것을 배울 수 있었습니다.** |
| **김나현** | **총 3개의 스테이지 level design 및 각 스테이지 별 블록 배치 기능 구현을 맡았다.**  **난생처음 들어본 nCurses 라이브러리 설치부터 리눅스/우분투 설치, 깃허브까지 온통 처음 보는 것들 뿐이라 프로젝트를 진행하는 내내 난항을 많이 겪었다. 그뿐만 아니라 c++에 대한 이해도와 코딩 실력이 부족하여 원래 맡기로 Item, Stage 클래스 구현도 하지 못하였다.**  **다행히 팀원의 배려로 그나마 snakegame에서 가장 쉬운 부분 구현을 맡을 수 있게 되었다. 신현우 학생이 만들어놓은 stage() 함수 뼈대 위에 각 스테이지 별로 벽을 배치하는 임무였는데, 해당 기능은 이중 반복문을 통해 구현하였다. 게임의 화면이 윈도 창 크기에 따라 유동적으로 변하는 것에 초점을 맞추어 어떤 크기의 창에서도 벽을 배치할 수 있게 만들려고 노력했다. 가장 쉬운 파트임에도 불구하고 초반과 후반 코드 수정 시 신현우 학생의 도움을 많이 받았다.**  **처음으로 진행하는 프로젝트에서 팀원에게 큰 도움이 되지 못하여 많이 아쉬웠지만, 다양한 툴 사용법도 익히고 팀원 간 소통의 중요성을 깨닫는 좋은 경험을 하게 된 것 같다.** |
|  |  |

# 참고 문헌

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처 | 발행년도 | 저자 | 기타 |
| 1 | 웹사이트 | https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/ |  |  |  |  |
| 2 | 웹사이트 | https://www.linuxjournal.com/content/about-ncurses-colors-0 |  |  |  |  |

# 부록

## 사용자 매뉴얼 및 설치 방법

1. **Ncurses 라이브러리를 설치합니다.**
2. **GitHub** [**https://github.com/plenilunel/2317sn**](https://github.com/plenilunel/2317sn) **의 repository를 복사합니다.**
3. **터미널을 켜고, 복사한 디렉토리로 이동한 후 해당 directory에서 아래 명령어를 타이핑합니다.**
   1. **make**
   2. **./main**

## 실행 영상

영상 촬영 및 편집 : 김나현

링크 : <https://www.youtube.com/watch?v=lrWQTw-J-bc>