**C++프로그래밍**

**프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *Snake game* |
| 학 번 | *20223100* |
| 이 름 | 심 준 |

|  |  |
| --- | --- |
| **Version** |  |
| **Date** | 05 |

**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

본 프로젝트는 C++ 언어와 ncurses 라이브러리를 이용하여 구현한 터미널 기반 스네이크 게임입니다. 사용자는 키보드를 통해 스네이크를 조작하며, 화면에 등장하는 아이템을 먹으며 점수를 획득하고, 특정 조건을 만족하면 미션이 완료되는 방식으로 구성되어 있습니다. 프로젝트의 목표는 전통적인 스네이크 게임의 재미를 유지하면서도, 미션 시스템, 게이트 이동, 성장 아이템/독 아이템 등의 요소를 추가하여 확장성과 난이도를 높이는 것이었습니다.

# 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 터미널 기반 C++ 스네이크 게임으로, 다양한 오브젝트(먹이, 독, 게이트)를 포함하고 있으며 각종 미션을 충족해야 클리어할 수 있도록 구성되어 있습니다. 주요 개발 목표와 방법은 다음과 같습니다:**

**• 게임 로직 구현**

**• Game 클래스에서 메인 루프 및 화면 업데이트, 키 입력, 충돌 처리 수행**

**• Snake 클래스는 방향 전환, 이동, 성장, 충돌 감지 기능 담당**

**• Map 클래스는 벽/빈 공간이 구성된 정적인 맵 구조를 관리**

**• 아이템 및 오브젝트 시스템**

**• Contents 클래스: 점수판 및 미션 달성 여부 표시**

**• Gate 클래스: 두 개의 벽 위치를 연결하여 순간이동 구현**

**• 먹이/독은 랜덤 위치에 생성되며, 이를 섭취 시 각각 성장/감소 효과 발생**

**• 미션 시스템 구현**

**• 사용자는 제한 시간 안에 길이 증가, 성장 아이템 획득 수, 게이트 통과 수 등 미션 조건을 달성해야 함**

### 시스템 구조 및 설계도

|  |  |  |
| --- | --- | --- |
| **파일명** | **클래스** | **설명** |
| main.cpp | 없음 | 게임 진입점. Game 클래스 실행 |
| game.hpp/cpp | Game | 전체 게임 루프, 상태 처리, 입력/출력 담당 |
| snake.hpp/cpp | Snake | 뱀의 위치 및 상태 관리, 이동/성장/충돌 기능 |
| map.hpp/cpp | Map | 정적 맵 구조 정의 및 제공 |
| contents.hpp/cpp | Contents | 점수판 및 미션 UI 출력 |
| gate.hpp/cpp | Gate | 게이트 위치 생성 및 텔레포트 기능 처리 |

**주요 클래스 구조 개요**

**• Game**

**• run(), process\_input(), update\_screen() 등 루프 제어**

**• Snake**

**텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**이 Snake 클래스는 스네이크 게임에서 뱀의 상태와 동작을 관리합니다.**

**• 속성:**

**• body: 뱀의 몸통 좌표들**

**• direction: 이동 방향**

**• length, score, alive: 길이, 점수, 생존 여부**

**• 기능:**

**• move(), grow(), shrink(): 이동, 성장, 축소**

**• check\_collision(): 충돌 감지**

**• get\_head(), get\_body(): 머리/몸 전체 좌표 반환**

**• set\_direction(), set\_alive(): 방향 및 상태 설정**

**• 특징:**

**• 기본값으로 초기화된 생성자**

**• 벡터를 이용한 유연한 몸통 관리**

**• 게임 로직과 연결되는 주요 클래스**

**• Map**

**• 2차원 배열 기반의 정적 맵 구조 제공**

**• Gate**

**• spawn(), teleport() 등 게이트 위치와 이동 로직 제공**

**spawn 함수는 게임 맵에서 두 개의 게이트를 무작위로 생성하는 역할을 합니다. 게이트는 반드시 벽(맵 데이터 값이 1) 위에 위치해야 하며, 두 개의 서로 다른 위치에 배치됩니다. 또한 생성된 게이트가 맵의 가장자리(위, 아래, 왼쪽, 오른쪽) 에 있는 경우, 이를 인식하여 진입 방향 정보를 함께 설정합니다.**

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**먼저 함수는 내부 변수 wall\_positions를 초기화하고, 맵 전체를 순회하여 벽(값이 1인 셀) 들의 좌표를 wall\_positions에 저장합니다. 이는 게이트를 만들 수 있는 후보 위치들입니다. 만약 벽이 2개 미만이면 게이트를 만들 수 없으므로 함수를 종료합니다.**

**그 다음, 무작위(random)로 두 개의 서로 다른 벽 좌표를 선택하여 게이트1과 게이트2의 위치로 사용합니다. 선택된 두 위치는 map 배열 상의 값을 7로 설정하여, 해당 좌표가 게이트임을 명시합니다. 그리고 exists = true로 현재 게이트가 존재함을 나타냅니다.**

**이후에는 각 게이트가 맵의 외곽(가장자리) 에 위치하는지를 확인합니다. 예를 들어, 행 좌표가 0이면 맵의 위쪽 벽, 열 좌표가 0이면 왼쪽 벽입니다. 가장자리에 위치해 있다면 is\_edge = true로 표시하고, 그 방향에 따라 edge\_dir 값을 설정합니다. 방향 정보는 이후 뱀이 게이트에 진입할 때 출구에서 어떤 방향으로 나올지를 결정하는 데 사용됩니다.**

**• Contents**

**• 현재 상태와 미션 달성 여부 출력**

### 활용/개발된 기술

사용한 기술 및 라이브러리

ncurses

• 터미널 UI 렌더링, 색상 설정, 실시간 키보드 입력 등 구현

C++ STL

• vector, pair 등 기본 자료구조로 좌표 및 오브젝트 상태 관리

• 게임 시간 지연 제어 및 미션 타이머 구현

개발한 알고리즘 및 기법

• 게이트 텔레포트 알고리즘

• 진입 방향에 따라 출구와 방향을 지정하여 부자연스러운 순간이동을 방지함

미션 UI 분리 처리

•미션 성공 여부를 Contents 클래스 내부에서 논리로 처리하고, 별도 출력 메서드로 분리함

### 현실적 제한 요소 및 그 해결 방안

|  |  |  |
| --- | --- | --- |
| **제한 요소** | **문제 내용** | **해결 방법** |
| 터미널 환경 종속 | 터미널 크기 부족 시 화면 깨짐 | 최소 해상도 조건 지정, ncurses 사용 |
| 게이트 출구 불확실성 | 진입 방향과 출구 방향 불일치로 버그 발생 가능 | teleport\_with\_dir() 함수로 방향 기반 출구 지정 |
| 사용자 입력 딜레이 | getch() 방식의 입력 지연 | nodelay() 및 논블록킹 키 입력 방식 사용 |
| 아이템 위치 충돌 | 먹이/독이 뱀이나 벽과 겹칠 수 있음 | 위치 재생성 시 유효성 검사 로직 추가 구현 |

### 결과물 목록

|  |  |
| --- | --- |
| **파일명** | **설명** |
| main.cpp | 게임 실행의 진입점으로, 인자 처리 및 Game 객체 실행 |
| game.hpp, game.cpp | 게임의 전반적인 흐름 관리 (루프, 입력, 출력, 상태 업데이트 등) |
| snake.hpp, snake.cpp | 뱀 객체의 상태(위치, 방향 등)와 동작(이동, 성장, 충돌 등) 처리 |
| map.hpp, map.cpp | 게임 맵 구조 정의 및 초기화, 맵 데이터 접근 인터페이스 제공 |
| contents.hpp, contents.cpp | 점수 및 미션 상태판 구현. 성장/포이즌/게이트 횟수와 미션 달성 여부 출력 |
| gate.hpp, gate.cpp | 게이트 생성 및 순간이동 처리. 진입·출구 게이트 위치 지정 |
| Makefile 또는 CMakeLists.txt (해당 시) | 전체 소스코드의 빌드 자동화 도구 파일 |
| README.md | 프로젝트 설명, 실행 방법, 키 조작법 안내 등 사용자 가이드 문서 |
| 맵 텍스트 파일들 (map.txt 등) | 맵 초기 구성을 위한 텍스트 기반 레이아웃 데이터 파일 (있는 경우) |
| 실행 바이너리 (snakegame 또는 main) | 컴파일 완료된 실행 파일. 콘솔에서 직접 실행 가능 |

# 자기평가

**게임의 기획부터 설계, 구현, 테스트 및 문서화까지 전 과정을 직접 수행하였습니다. 콘솔 환경에서 동작하는 스네이크 게임을 목표로 하였고, 객체지향적으로 클래스를 분리하여 구조적인 코드를 작성하려 노력했습니다.**

**개발 초반에는 전체 시스템 구조를 어떻게 나눌지 고민이 많았고, 특히 Snake, Map, Gate, Game 클래스 간의 의존성과 역할 분담을 정리하는 데 시간이 많이 소요되었습니다. 다행히 GitHub Copilot을 통해 반복적인 코드 패턴이나 함수 정의 등을 빠르게 제안받을 수 있었고, 구현 속도를 높이는 데 많은 도움이 되었습니다. 단, Copilot이 제안하는 코드가 항상 프로젝트 구조에 적합하지는 않았기 때문에 이를 직접 수정하고 문맥에 맞게 통합하는 작업이 필요했습니다.**

**개인적으로 가장 어려웠던 부분은 게이트 시스템 구현이었습니다. 뱀이 진입하는 방향과 출구에서의 방향 전환을 자연스럽게 구현하려고 노력했으며, 이 과정에서 좌표 및 방향 정보를 함께 처리하는 알고리즘을 직접 설계했습니다.**

**프로젝트 전반적으로 만족스러운 결과를 얻었지만, 이후 유지보수성과 테스트 편의성을 고려한다면 클래스 간 인터페이스를 좀 더 명확히 하고, 단위 테스트 도구를 도입하는 개선이 필요하다고 생각합니다.**

**이번 프로젝트는 C++ 및 ncurses 사용 경험을 실전적으로 쌓을 수 있는 기회였으며, GitHub Copilot을 단순한 보조 도구가 아닌 개발 방향을 고민하는 참고자료로 적절히 활용할 수 있게 된 점이 의미 있었습니다.**

# 참고 문헌

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처 | 발행년도 | 저자 | 기타 |
|  | 서적 |  |  |  |  |  |
|  | 기사 |  |  |  |  |  |

# 부록

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

## 사용자 매뉴얼

**프로젝트 실행 후 사용안내, 따라하기 등 포함**

## 설치 방법

**프로그램 컴파일 및 실행 방법 포함**