**2024-1학기 소프트웨어개발도구및환경실습 최종 프로젝트**

CSE 20231609 정희선

1. **프로젝트 목표**

최종 프로젝트의 목표는 지뢰 찾기 프로그램을 만드는 것이다. 지뢰 찾기 프로그램은 화면에 지뢰 필드를 그리는 것, 사용자의 마우스 입력과 상호작용하는 것, 지뢰 찾기 게임 로직을 처리하는 것, 게임 진행 상태에 대한 피드백을 제공하는 것으로 구성된다.

1. **실험 환경에 대한 설명**

지뢰 찾기 프로젝트는 Microsoft Visual Studio 2022와 OpenFrameworks를 활용하여 개발되었다. OpenFrameworks는 C++ 기반의 오픈 소스 툴킷으로, 인터랙티브 미디어 애플리케이션 개발에 적합한 다양한 기능을 제공한다. 이 환경을 통해 지뢰 찾기 게임의 그래픽 처리와 사용자 입력 반응이 처리된다. OpenFrameworks의 그래픽 라이브러리는 게임 보드와 타일의 상태를 명확하게 시각화하고 이벤트 시스템은 사용자의 클릭에 반응하여 게임의 로직을 실행한다.

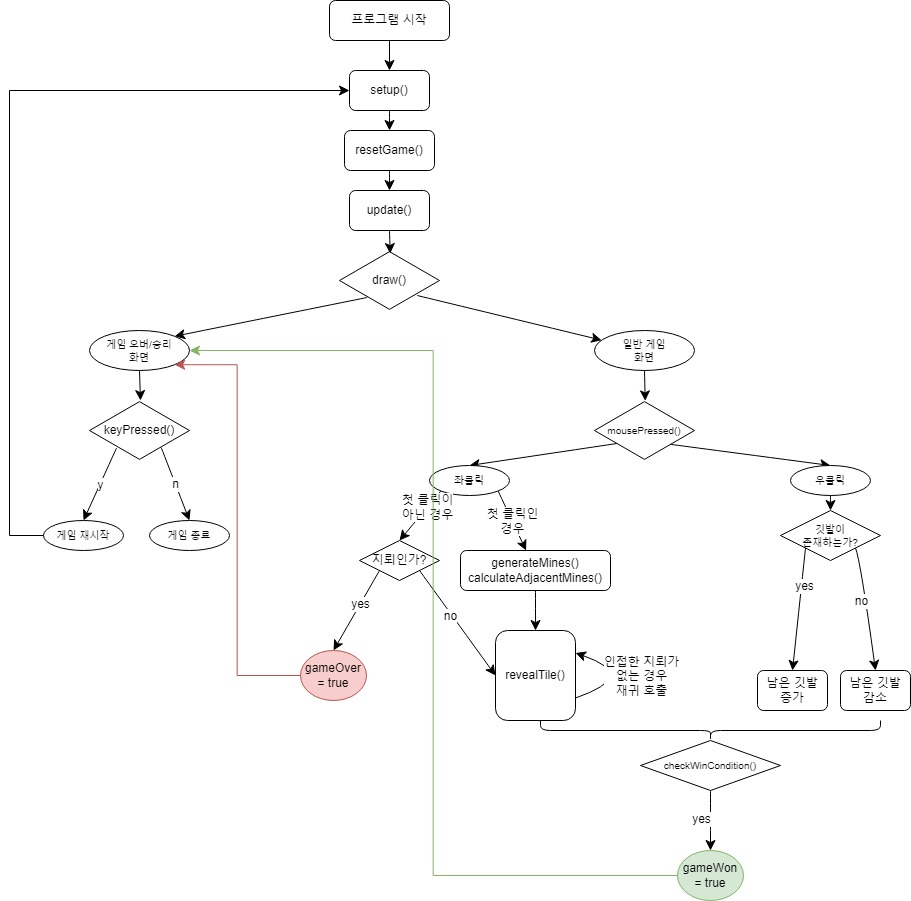
1. **각 변수에 대한 설명**

|  |  |
| --- | --- |
| **int rows;**  **int cols;** | 게임 보드의 행(열) 수를 저장하고 보드의 세로(가로) 크기를 결정한다. |
| **vector<vector<bool>> mines;** | 각 칸에 지뢰가 있는지 여부를 저장하는 2차원 벡터이다. true면 지뢰가 있음을 나타낸다. |
| **vector<vector<bool>> revealed;** | 각 칸이 사용자에게 공개되었는지 여부를 저장하는 2차원 벡터이다. true면 해당 칸이 공개되었음을 의미한다. |
| **vector<vector<bool>> flagged;** | 각 칸에 깃발이 표시되었는지 여부를 저장하는 2차원 벡터이다. 깃발은 지뢰가 있을 것으로 예상되는 칸에 사용자가 표시할 수 있다. |
| **vector<vector<int>> adjacentMines;** | 각 칸 주변에 있는 지뢰의 수를 저장하는 2차원 벡터이다. 칸을 클릭했을 때 주변의 안전한 영역을 판단하는 데 사용된다. |
| **bool gameOver;**  **bool gameWon;** | 게임 오버(게임 승리) 상태를 나타낸다. gameOver 변수가 true면 게임이 끝났음을 의미하고, gameWon 변수가 true면 사용자가 게임에서 승리했음을 의미한다. |
| **bool isFirstClick;** | 사용자의 첫 클릭인지를 확인하는 플래그이다. 첫 클릭에서는 지뢰가 배치되기 전이므로 지뢰를 피할 수 있다. |
| **int flagsLeft;** | 사용할 수 있는 깃발의 남은 수를 나타낸다. 이 숫자는 게임의 진행 상황에 따라 감소하거나 증가할 수 있다. |

1. **각 함수에 대한 설명**

|  |  |
| --- | --- |
| **void setup();** | **역할 :** 프로그램 시작 시 초기 설정을 수행하는 함수이다. 게임 보드의 크기를 설정하고 게임의 초기 상태를 준비한다. |
| **void update();** | **역할 :** 게임의 상태를 갱신하는 함수이다. 게임 로직에 따라 내부 상태를 업데이트하는 데 사용된다. |
| **void draw();** | **역할 :** 게임 화면을 그리는 함수이다. 보드의 현재 상태를 그래픽으로 표현하며, 게임 오버나 승리 메시지를 사용자에게 보여준다. |
| **void keyPressed(int key);** | **역할 :** 키보드 입력을 처리하는 함수이다. 사용자가 키를 누를 때 호출되며, 게임의 흐름을 제어하는 데 사용된다.  **파라미터 :**  int key - 사용자가 누른 키를 나타내는 정수 값이다. |
| **void mousePressed(int x, int y, int button);** | **역할 :** 마우스 클릭을 처리하는 함수이다. 사용자가 마우스로 게임 보드의 특정 타일을 클릭할 때 호출되어 해당 타일의 상태를 변경한다.  **파라미터 :**  int x - 클릭된 위치의 x 좌표  int y - 클릭된 위치의 y 좌표  int button - 클릭된 마우스 버튼(왼쪽, 오른쪽) |
| **void generateMines(int mineCount, int firstClickRow, int firstClickCol);** | **역할 :** 지뢰를 무작위로 배치하는 함수이다. 사용자의 첫 클릭 위치를 제외한 게임 보드에 지정된 수의 지뢰를 배치한다.  **파라미터 :**  int mineCount - 배치할 지뢰의 수  int firstClickRow - 첫 클릭이 일어난 타일의 행  int firstClickCol - 첫 클릭이 일어난 타일의 열 |
| **void calculateAdjacentMines();** | **역할 :** 각 타일 주변의 지뢰 수를 계산하는 함수이다. 게임 보드의 각 타일에 대해 인접한 여덟 칸에 있는 지뢰의 수를 계산하여 저장한다. |
| **bool isValid(int row, int col);** | **역할 :** 주어진 타일의 위치가 게임 보드 내에 유효한지 검사하는 함수이다.  **파라미터 :**  int row - 검사하려는 타일의 행  int col - 검사하려는 타일의 열  **반환값** : 타일 위치가 유효하면 true, 그렇지 않으면 false |
| **void revealTile(int row, int col);** | **역할 :** 타일을 공개하는 함수이다. 사용자가 타일을 클릭했을 때 해당 타일과 인접한 지역을 재귀적으로 공개한다. 인접한 지뢰가 없는 타일을 클릭하면 주변 타일도 연쇄적으로 공개된다. **파라미터 :**  int row - 공개하려는 타일의 행  int col - 공개하려는 타일의 열 |
| **void checkWinCondition();** | **역할 :** 게임 승리 조건을 검사하는 함수이다. 모든 지뢰가 아닌 타일이 공개되었을 때 승리 상태로 전환한다. |
| **void drawTiles();** | **역할 :** 게임 보드의 각 타일을 그리는 함수이다. 타일의 상태(공개, 비공개, 깃발 표시)에 따라 다르게 그린다. |

1. **프로젝트 전체 플로우 차트**

****

1. **자료구조 및 알고리즘**
2. **자료구조**

* **2차원 벡터 배열**

mines, revealed, flagged: 각각 지뢰의 위치, 칸의 공개 여부, 깃발의 위치를 저장하는 데 사용된다. 각 배열은 bool 타입의 값으로 구성되어 있으며, 게임 보드의 상태를 표현한다.

adjacentMines: 각 칸에 인접한 지뢰의 수를 저장한다. 이는 int 타입의 2차원 배열로, 사용자에게 해당 칸 주변의 위험도를 시각적으로 표시하는 데 사용된다.

1. **알고리즘**

* **지뢰 생성 알고리즘**

사용자가 첫 번째 클릭을 한 위치를 기준으로 주변에 지뢰가 생성되지 않도록 보장해 첫 클릭에 게임이 끝나는 것을 방지한다. 지뢰는 랜덤 위치에 생성되되, 첫 클릭 주변은 제외한다.

* **타일 공개 알고리즘 (백트래킹 알고리즘)**

기본 조건 설정: 타일이 유효한 위치에 있는지 확인하고, 이미 공개되었거나 지뢰가 있는 타일이라면 더 이상 진행하지 않는다. 이는 재귀의 기본 조건으로 작동하여 불필요한 탐색을 줄인다.

타일 공개: 유효한 타일이면 해당 타일을 공개한다. 이는 사용자에게 해당 위치의 지뢰 유무 정보를 제공하며, 인접한 지뢰의 수를 표시한다.

인접 지뢰 수 확인: 공개된 타일 주변의 지뢰 수가 0이면, 해당 타일은 주변에 지뢰가 없다는 것을 의미한다. 이 경우, 안전하다고 판단되는 주변 타일들을 자동으로 공개하기 위해 재귀 호출을 사용한다.

재귀적 탐색: 현재 타일에서 인접한 모든 타일에 대해 revealTile 함수를 재귀적으로 호출한다. 이 과정에서 각 타일은 다시 위의 조건들을 검토하게 되며, 안전한 타일이 계속해서 연쇄적으로 공개된다.

백트래킹 실행: 만약 주변 타일 공개 과정에서 지뢰가 있는 타일을 만나면 재귀적 호출은 그 지점에서 멈춘다. 이는 백트래킹의 특징을 보여준다: 가능한 모든 경로를 탐색하지만, 문제의 조건에 맞지 않는 경로는 더 이상 진행하지 않고 되돌아간다.

1. **시간/공간 복잡도**
2. 시간복잡도

revealTile 함수: 재귀적으로 호출되며, 주변 타일들이 안전하다면 계속해서 인접한 타일을 공개한다. 최악의 경우, 모든 타일이 연결되어 있고 각 타일을 최소 한 번씩 방문해야 한다. 이러한 경우 시간복잡도는 O(N×M)이 된다. (N은 행의 수, M은 열의 수) 각 타일마다 인접한 8개 타일을 확인하므로, 각 함수 호출당 최대 8번의 추가 호출이 발생할 수 있다. 그러나 실제 게임에서는 모든 타일이 서로 연결되지 않기 때문에 실제 시간복잡도는 훨씬 낮다.

calculateAdjacentMines 함수: 게임 보드의 모든 타일을 한 번씩 방문하며 각 타일에 대해 8개의 인접 타일을 검사한다. 따라서 O(N×M)의 시간복잡도를 가진다.

generateMines 함수: 주어진 지뢰 수만큼 반복한다. 각 지뢰의 배치는 상수 시간이 걸리며, 최악의 경우 모든 타일을 검토한다. 그러나 평균적으로 이 함수의 시간복잡도는 지뢰 수에 비례하므로 O(M)이다. (M은 지뢰의 수)

1. 공간복잡도

게임 보드 저장: mines, revealed, flagged, adjacentMines 배열은 모두 N×M 크기의 공간을 사용한다. 이는 각 타일의 상태를 저장하기 위해 필요한 공간이므로 전체 공간복잡도는 O(N×M)이다.

재귀 호출 스택: revealTile 함수는 재귀적으로 호출될 수 있으며, 호출 스택의 최대 깊이는 타일의 최대 수인 N×M이다. 따라서 이 부분의 공간복잡도도 O(N×M)이다.

결론적으로, 지뢰찾기 게임의 시간복잡도와 공간복잡도는 모두 O(N×M)이다.

1. **창의적 구현 항목에 대한 설명**

지뢰 찾기 프로젝트를 통해 수업 시간에 배운 DFS 알고리즘의 한계를 극복하고 더 효과적인 탐색을 가능하게 하는 백트래킹 기법에 주목했다. 백트래킹은 지뢰 찾기 게임에서 타일을 클릭했을 때, 해당 타일이 지뢰가 아니고 주변에 지뢰가 없다면 연결된 모든 안전한 타일을 자동으로 열어주는 데 사용된다. 이 과정에서 백트래킹은 재귀적으로 각 타일을 확인하며, 이미 열린 타일이나 지뢰가 있는 타일은 건너뛰는 방식으로 작동한다.

1. **프로젝트 실행 결과**

|  |  |
| --- | --- |
|  | 시작 화면 |
|  | 첫 클릭 후 화면 |
|  | 게임 실행 중 화면 |
|  | 게임 오버 화면 |
|  | 게임 승리 화면 |

1. **느낀 점 및 개선 사항**

백트래킹 알고리즘 등 다양한 알고리즘으로 지뢰 찾기 게임을 구현하며, 평소에 좋아하는 게임의 로직을 깊이 있게 이해할 수 있었다.

지금은 10\*10 그리드의 지뢰 찾기를 구현했지만 더 큰 그리드에서의 게임도 구현하고 싶다. 또 그렇게 만든 더 큰 그리드의 지뢰 찾기 게임을 가지고 사용자가 게임의 난이도를 선택할 수 있도록 설정을 추가하여, 초보자부터 고급 사용자까지 즐길 수 있는 지뢰 찾기 게임을 구현해보고 싶다.