**최종프로젝트 보고서**

**전공: 컴퓨터공학과 학년: 2학년 학번: 20231632 이름: Jumagul Alua**

**1. 프로젝트 목표**

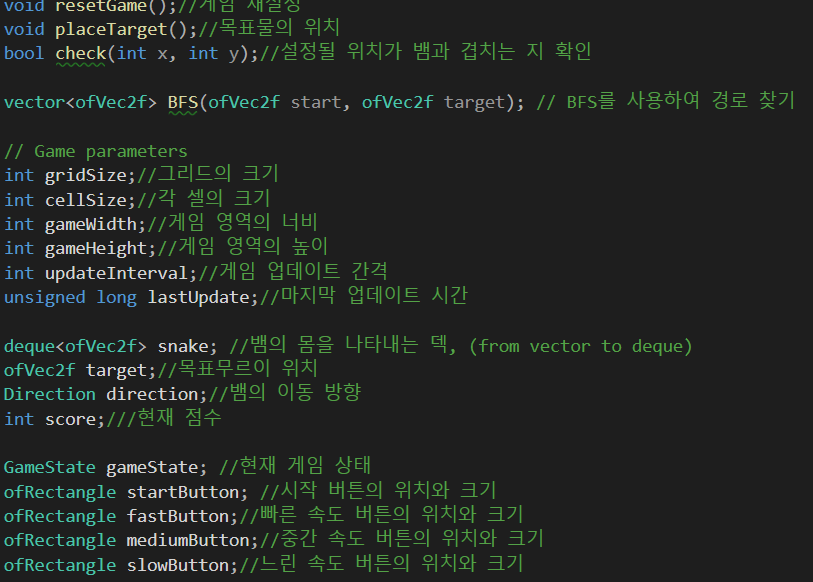
모두가 아는 ‘Snake game’을 OpenFrameworks 오픈 소스를 활용해 C++ 언어로 작성해본 프로젝트이다. 이 게임에서는 뱀이 주어진 목표물(음식)을 먹으면서 성장한다, 목표물에 닿을 때마다 점수가 증가한다. 경로 탐색 알고리즘으로 BFS (Breadth-First Search)를 사용하여 뱀이 목표물에 도달할 수 있도록 경로를 계산해서 자기 알아서 게임 한. 또한, 뱀의 속도를 조절할 수 있는 기능을 추가하여 빠른 속도, 중간 속도, 느린 속도 중에서 선택할 수 있다.

**2. 실험 환경에 대한 설명**

* 하드웨어: Lenovo 노트북 모델 82HS
* 운영체제: Windows 11
* 개발 도구: Visual Studio 2019 version 16.11.37
* 프레임워크: OpenFrameworks 0.11.0

코드는 Visual Studio 2019에서 작성되었으며, OpenFrameworks 프로젝트 생성기를 통해 기본 프로젝트를 만들고, ofApp.cpp, ofApp.h, main.cpp 파일에 작성하여 실행하였다.

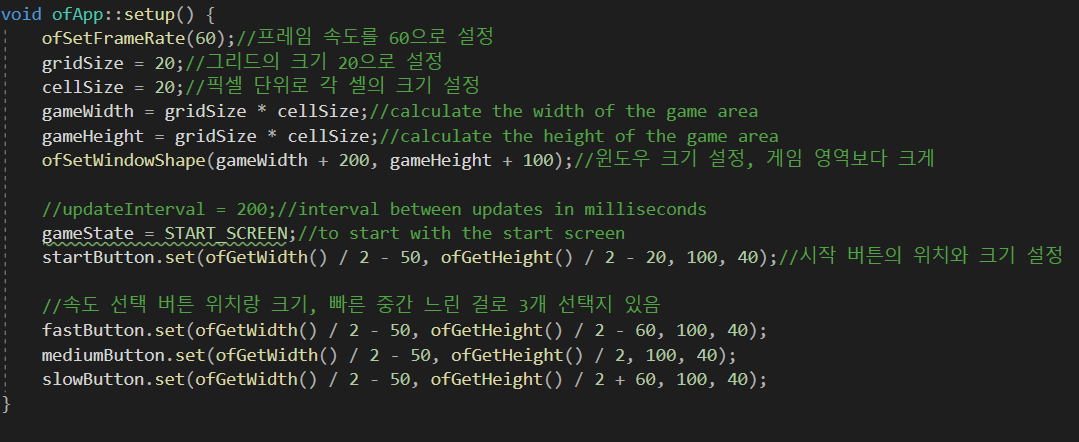
**3. 각 변수에 대한 설명**



여기서 ofApp.h로 각 변수를 설명해본다.

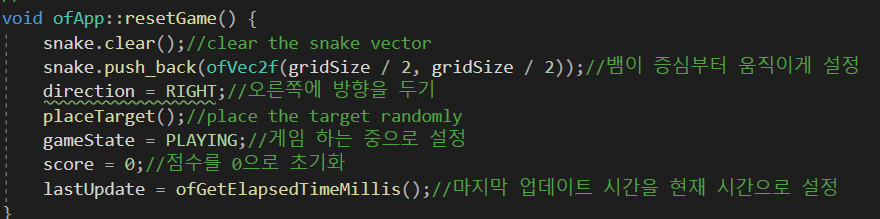
* gridSize: 게임의 그리드 크기를 결정한다. 그리드의 가로와 세로의 칸 수를 정의하며, 전체 게임 영역을 설정하는데 사용된다.
* cellSize: 각 그리드 셀의 크기를 픽셀 단위로 설정한다.
* gameWidth, gameHeight: 게임 영역의 너비와 높이다. gridSize와 cellSize를 곱하여 계산된다.
* updateInterval: 게임 업데이트 간격이다. 밀리초 단위로 설정되며, 게임의 속도를 결정한다.
* lastUpdate: 마지막으로 게임 상태를 업데이트한 시간을 저장한다.
* snake: 뱀의 몸을 나타내는 ofVec2f 객체의 덱이다. 각 ofVec2f 객체는 뱀의 각 부분의 위치를 나타낸다.
* target: 뱀이 먹어야 하는 목표물의 위치이다.
* direction: 뱀의 현재 이동 방향을 나타낸다.
* score: 현재 게임의 점수이다.
* gameState: 현재 게임의 상태를 나타내며, 게임의 흐름을 제어한다.
* startButton, fastButton, mediumButton, slowButton: 시작 및 속도 선택 버튼의 위치와 크기를 나타내는 사각형이다.

**4. 각 함수에 대한 설명**



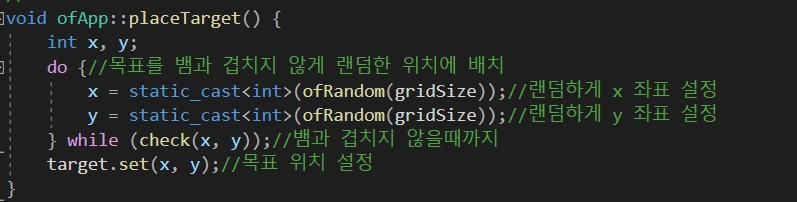
게임 초기 설정을 담당하며, 화면 크기, 게임 그리드, 셀 크기, 버튼 위치 등을 설정하는 setup 함수이다. 일단, 게임의 프레임 속도를 60으로 설정하여 부드러운 움직임을 보장해준다.gridSize로 게임 그리드의 크기를 설정하며, 뱀과 목표물이 위치할 수 있는 공간을 결정한다. 그리드의 각 칸은 gridSize로 정의되며, 이 값은 뱀의 움직임 단위를 결정한다. 그리드의 각 셀의 크기를 픽셀 단위로 설정하여 화면에 그려질 때의 크기를 결정하는 cellSize를 성정해준다. gameWidth, gameHeight로 게임 화면의 너비와 높이를 정의한다. gridSize와 cellSize를 곱하여 게임 영역의 전체 크기를 계산한다. 그 다음, ofSetWindowShape로 전체 윈도우의 크기를 설정한다. 게임 영역보다 크게 설정하여 주변 여백과 UI 요소를 포함할 수 있도록 한다. gameState는 초기 화면 상태를 START\_SCREEN으로 설정하여 게임 시작 시 화면에 시작 버튼이 표시되도록 한다. 시작 버튼 및 속도 선택 버튼의 위치와 크기를 설정하여 화면에 배치하기 위해 startButton, fastButton, mediumButton, slowButton를 선언해준다. ofRectangle 자료구조를 사용하여 버튼의 위치와 크기를 저장한다.

-----------------------------------------------------------------------------------------------------------------



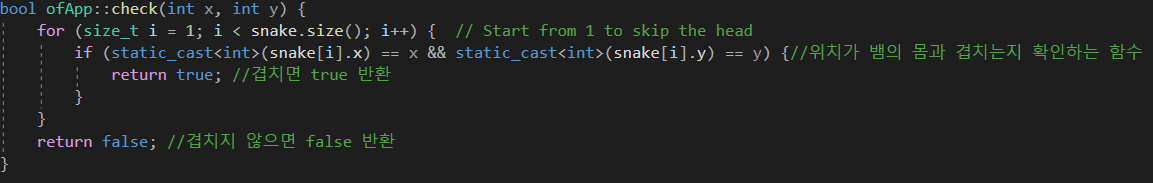
게임을 재설정하여 새로운 게임을 시작할 때 초기화 작업을 수행한다. 뱀의 위치, 목표물, 방향, 점수 등을 초기화한다. 뱀의 몸을 저장하는 deque<ofVec2f> 자료구조를 초기화한다. 덱(deque)을 사용해 양쪽에서 삽입과 삭제고, 뱀의 머리와 꼬리를 빠르게 추가 및 제거할 수 있다. ofVec2f는 2D 벡터를 나타내며, 뱀의 각 부분의 위치를 저장한다.OfVec2f 사용해서 그리드의 너비와 넓이를 2로 나누므로서 뱀의 시작 위치를 그리드 중앙에 배치한다. 뱀의 초기 방향을 enum 타입을 사용해 오른쪽으로 설정해주고 뱀의 이동 방향을 결정해주다. 그리고 목표물을 랜덤한 위치에 설치해준 다음, 게임 상태를 게임 중임으로 설정한다. 점수를 0으로 초기화한다. lastUpdate로 게임의 첫 업데이트 시점을 기준으로 시간 차이를 계산할 수 있게 한다.

-----------------------------------------------------------------------------------------------------------------



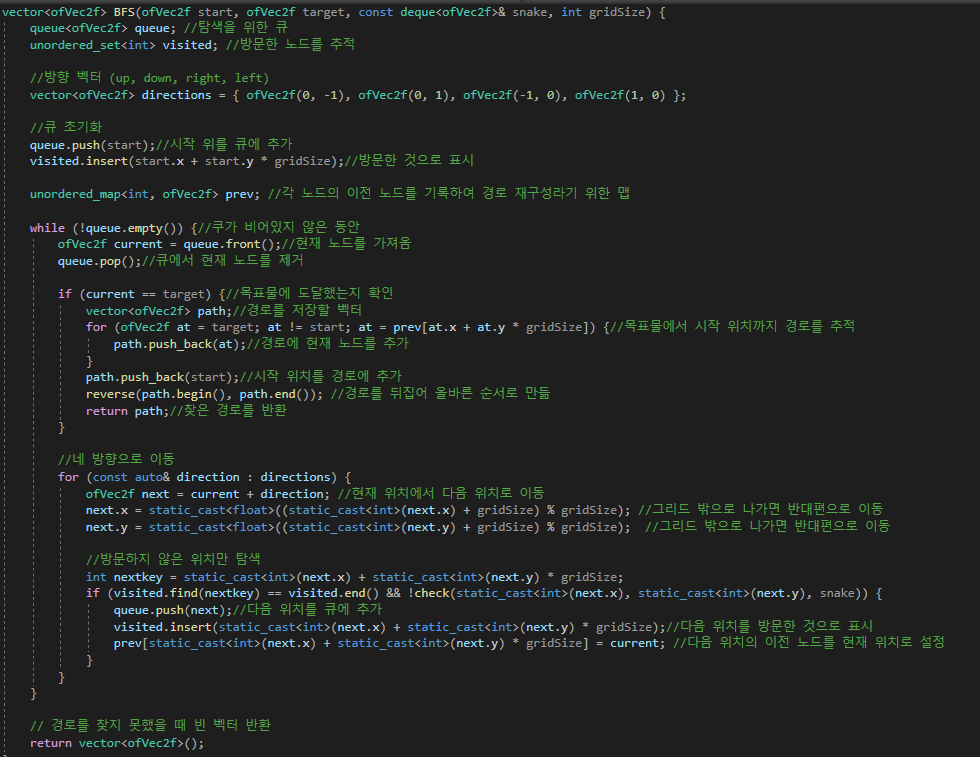
물표물을 뱀의 몸과 겹치지 않는 랜덤한 위치에 배치하는 함수이다. Check 함수를 통해 뱀과 겹치지 않을 것을 확인하면서 ofRandom으로 랜덤하게 x, y 좌표를 설정해준다. target.set으로 ofVec2f 자료구조를 사용하여 2D 벡터로 위치를 저장해준다.

-----------------------------------------------------------------------------------------------------------------



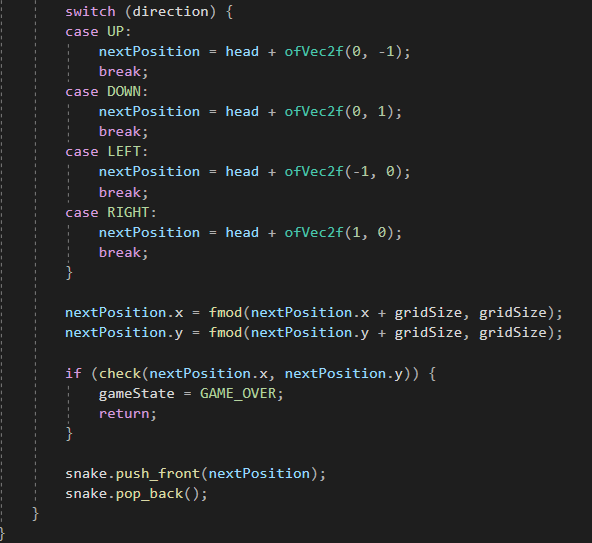
기어진 위치가 뱀의 몸과 겹치는지 확인하는 함수이다. for 루프 안에는 뱀의 각 부분을 순회하며 주어진 위치와 겹치는지 확인하고 만약 주어진 x, y 좌표와 뱀의 부분의 좌표가 겹치지 않는다면 false, 동일하면 true를 반환한다.

-----------------------------------------------------------------------------------------------------------------



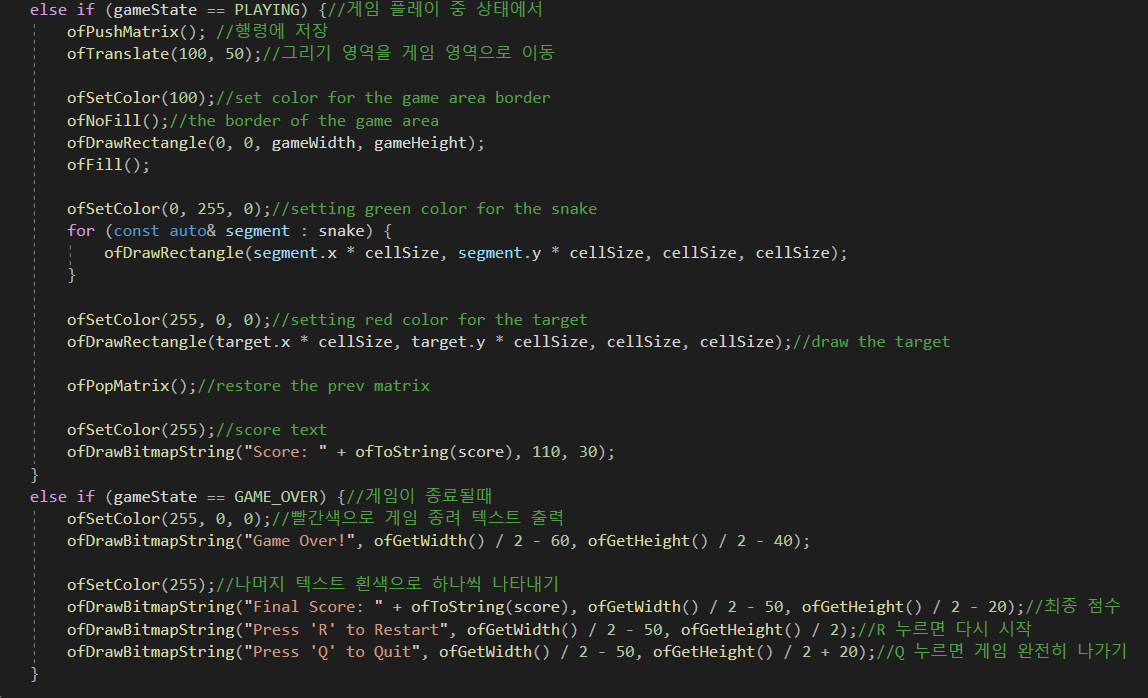
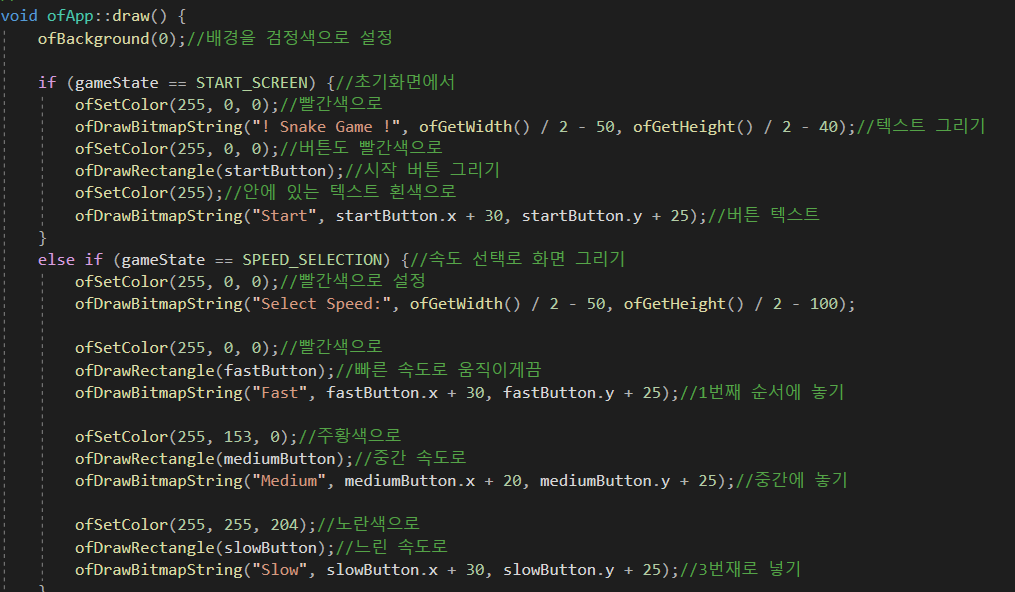
시작 위치에서 목표물까지의 최단 경로를 찾기 위해 수업 시간에 배운 BFS 알고리즘을 사용해봤다. 2D 그리드에서 이동하며, 그리드의 경계를 넘어갈 경우 반대편으로 이동하도록 처리된다. 또한, 뱀 게임의 뱀 몸통과 겹치지 않도록 경로를 찾아야 한다. queue<ofVec2f> queue는 BFS 탐색을 위해 사용할 큐로, 탐색할 각 위치를 저장한다. 큐는 FIFO 방식으로, 먼저 들어온 노드를 먼저 처리한다. unordered\_set<int> visited: 방문한 노드를 추적하기 위한 집합으로, 각 위치를 정수형 키로 변환하여 빠르게 검색하고 추가할 수 있다. 시작 위치를 큐에 추가하여 탐색을 시작한다. 시작 위치를 방문한 것으로 표시하고 위치를 정수형으로 변환하여 저장한다. while 루프로 큐가 비어있지 않은 동안 큐의 각 노드를 처리하여 목표 위치까지의 경로를 찾는다. 현재 위치에서 다음 위치로 이동하면서 그리드 밖으로 나가면 반대편으로 이동하도록 x와 y좌표를 조정해준다. 만약 다음 위치가 방문되지 않았고, 뱀의 몸과 겹치지 않는다면, 다음 위치를 큐에 추가하여 탐색을 계속한다. 다음 위치를 방문한 것으로 표시하고 다음 위치의 이전 노드를 현재 위치로 설정하여 경로를 재구성할 수 있도록 한다. Bfs를 이용해 뱀이 목표까지 최대 근처의 있는 길로 찾아주면서 가준다.

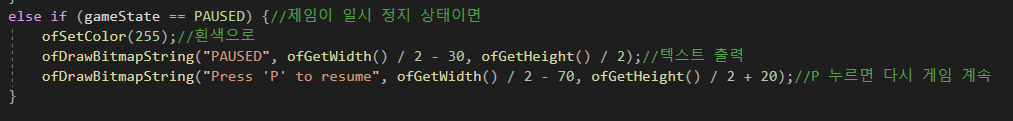
-----------------------------------------------------------------------------------------------------------------



게임 상태를 업데이트하여 뱀의 위치를 갱신하고, BFS를 사용하여 목표물까지의 경로를 계산하여 그 경로를 따라 이동하게 하는 함수이다, 현재 게임 상태가 PLAYING인지 확인한다. ofGetElapsedTimeMillis()를 통해 현재 시간을 가져온다. 마지막 업데이트 시간을 사용하여 현재 시간과의 차이를 계산해 보고 현재 시간으로 갱신한다. BFS를 사용하여 뱀의 머리에서 목표물까지의 경로를 찾는다. BFS로 찾은 경로를 저장하고 만약 경로가 비어있지 않다면 경로의 다음 위치로 이동하게 한다. 경로의 다음 위치를 가져온다, 여기서 [0]은 현재 위치이므로 [1]이 다음 위치인다. 그리고 뱀의 머리를 새로운 위치로 이동한다. 만약 뱀이 목표물에 도달했다면 새로운 목표물을 배치하고 점수를 증가시키고, 도달하지 않았다면 뱀의 꼬리를 제거하여 이동을 반영한다. 그리고 마지막으로 check 함수로 새로운 위치가 뱀의 몸과 겹치는지 확인하여 겹친다면 게임 상태를 GAME\_OVER로 설정합니다. 그렇지 않으면 뱀의 머리를 다음 위치로 이동시키고, 목표물을 먹었는지 확인하여 점수를 올리거나, 목표물을 먹지 않은 경우 뱀의 꼬리를 제거한다. 경로를 찾지 못했을 때는 현재 방향으로 뱀의 머리를 이동시킨다. 이동한 위치가 뱀의 몸과 겹치는지 확인하고, 겹치는 경우 게임 오버 상태로 전환합니다. 그렇지 않으면 머리를 다음 위치로 이동시키고, 꼬리를 제거합니다.

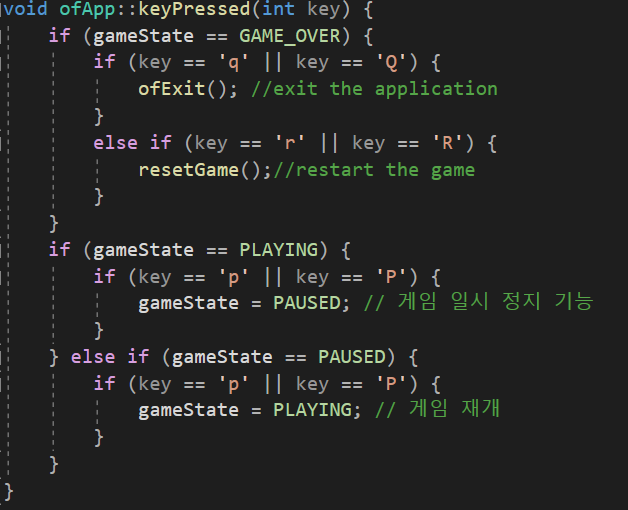
-----------------------------------------------------------------------------------------------------------------





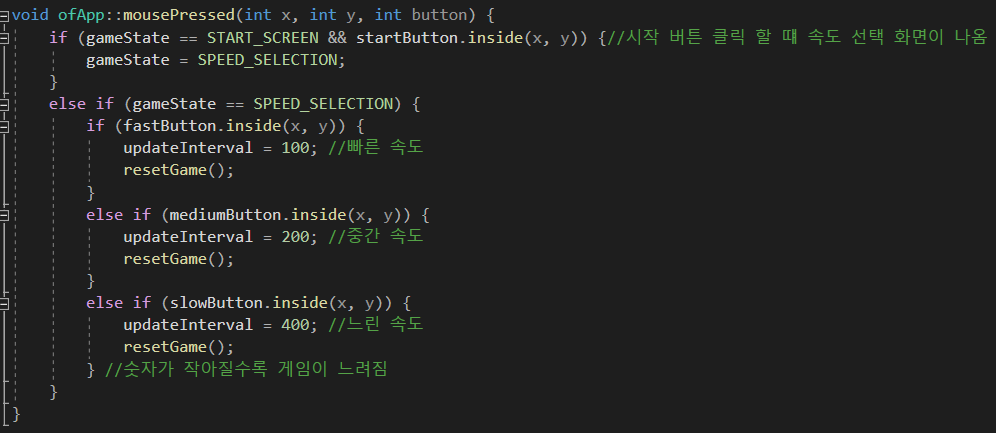
현재 게임 상태에 따라 화면에 그리기 작업을 수행하는 함수이다. 여기는 게임 상태에 때라 시작 화면, 속도 선택 화면, 플레이 중 화면, 게임 오버 화면을 각각 그려주는 걸로 되어 있다. 일단, 배경을 검정색으로 설정한다. 게임 상태가 시작 화면일 때 시작 버튼과 텍스트를 그려준다. 게임 상태가 속도 선택 화면일 때 속도 선택 버튼과 텍스트를 그린다. ofDrawRectangle를 이용해 빠른 속도, 중간 속도, 느린 속도의 버튼을 그린다. 게임 상태가 플레이 중일 때 뱀, 목표물, 점수를 그란다. 게임 상태가 게임 오버일 때 게임 오버 텍스트와 버튼을 그리고, 일지 정지 일때도, 정지인 것을 나타내는 텍스트를 보여준다.

-----------------------------------------------------------------------------------------------------------------



키 입력을 처리하여 뱀의 방향을 변경하거나 게임 상태에 따라 동작을 수행하는 함수이다. ‘Q' 키를 누르면 프로그램을 종료하고, 'R' 키를 누르면 게임을 재시작하고 ‘P’ 키를 누르면 게임이 일시 정지 되면 다시 누를때 다시 게임 시작된다.

-----------------------------------------------------------------------------------------------------------------



마우스 클릭을 처리하여 시작 버튼을 클릭하거나 속도 선택 버튼을 클릭할 때 동작을 수행하는 함수이다. 게임 상태가 시작 화면이고 시작 버튼을 클릭했을 때 속도 선택 화면으로 전환한다. 빠른 속도 버튼을 클릭했을 때 업데이트 간격을 100밀리초로, 중간 속도는 간격인 경우 200밀리초로 설정하고 느린 속도 버튼 경우, 간격을 400밀리초로 설정하고 게임을 재시작한다.

**5. 프로젝트 전체 플로우차트 + 자료구조 및 알고리즘 + 시간/공간 복잡도 상세히 설명.**

### **플로우차트:**

### 프로그램 시작 시 setup() 함수 호출 -> START\_SCREEN 상태에서 시작 버튼을 누르면 SPEED\_SELECTION 상태로 전환-> 속도를 선택하면 PLAYING 상태로 전환되고 게임이 시작->update() 함수에서 뱀의 위치를 업데이트하고, 목표물에 도달하면 점수를 증가->뱀이 자신의 몸과 충돌하면 GAME\_OVER 상태로 전환->게임 오버 상태에서 Q를 누르면 프로그램 종료, R을 누르면 게임 재시작, P를 누르면 일시정지

### **자료구조:**

* deque<ofVec2f> snake: 뱀의 몸을 저장하는 덱, 뱀의 머리와 꼬리를 쉽게 추가 및 제거할 수 있다.
* ofVec2f target: 목표물의 위치를 저장하는 2차원 벡터.
* unordered\_map<int, ofVec2f> prev: BFS에서 경로를 재구성하기 위해 사용한 이전 노드를 저장하는 맵.
* unordered\_set<int> visited: BFS에서 방문한 노드를 추적하기 위한 집합.
* queue<ofVec2f>: BFS에서 탐색을 위한 큐.

**알고리즘**:

* BFS: 최단 경로를 찾기 위한 알고리즘으로, 시작 위치에서 목표물까지의 경로를 계산한다.
* 경로 탐색: 뱀이 목표물로 도달할 수 있는 경로를 BFS를 통해 찾고, 해당 경로를 따라 이동한다.

### **시간/공간 복잡도**

* **시간 복잡도**:

-BFS: O(V+E)(V는 그리드의 노드 수, E는 간선 수). 그리드에서 모든 셀을 탐색해야 하므로 최악의 경우 O(N^2)가 된다.

-update(): BFS 호출에 의한 시간 복잡도는 O(N^2)이다.

* **공간 복잡도**:

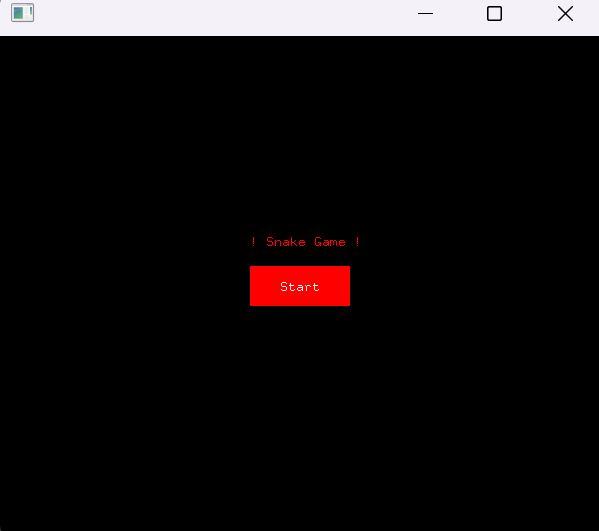
-BFS: O(V), BFS에서 사용되는 자료구조의 공간 복잡도이다.

-전체 게임: O(N^2), 그리드 크기와 BFS에 의해 사용되는 자료구조로 인해 O(N^2) 공간이 필요한다.

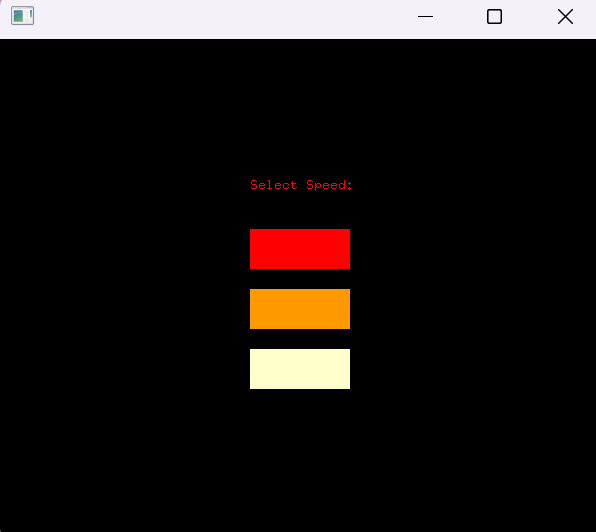
**6. 창의적 구현 항목에 대한 설명**

* 속도 선택 기능: 게임 시작 전에 사용자가 뱀의 속도를 선택할 수 있도록 하여 게임의 난이도를 조절할 수 있다. 이를 통해 사용자 경험을 향상시킬 수 있다.
* BFS를 이용한 경로 탐색: 뱀이 목표물까지 도달할 수 있도록 BFS 알고리즘을 활용하여 경로를 계산한다.

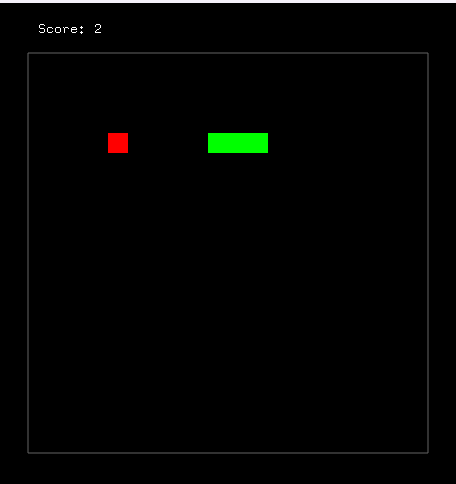
**7. 프로젝트 실행 결과 캡쳐**



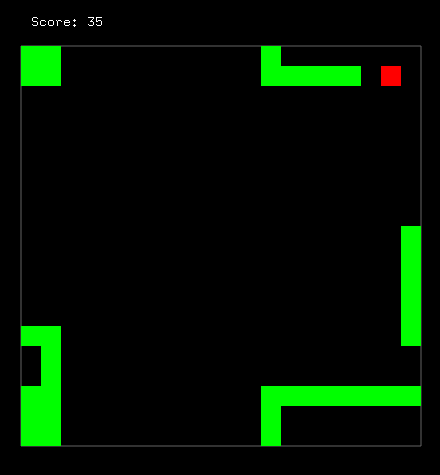
초기 화면(시작 버튼을 눌러 시작하기)



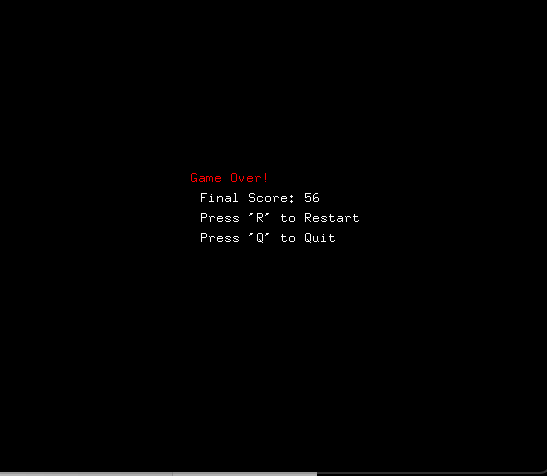
속도를 선택하기 위한 화면, 빨간 색은 빠른 속도, 주황색은 중간 속도, 노란색은 느린 속도이디,



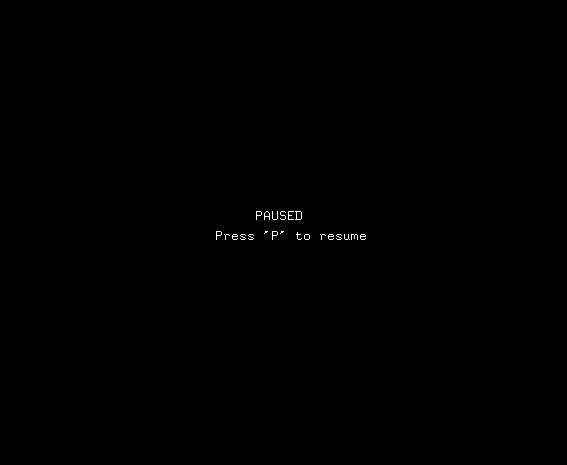
플레이 중일때



Bfs를 사용하기 떄문에, 목표물에 이동하기 위해서 더 가까이 있는 것을 가준다.



자기 자신과 겹칠때, 게임 종료 및 최종 점수 출력. 재시작을 원하면, R를 누르라는 텍스트와 게임 나가기 위한 Q를 눌러야되는 텍스트 출력



P를 누렀을때, 일시 정지 화면

**8. 느낀 점 및 개선 사항**

이번 프로젝트를 통해, 지금까지 배운 C와 OpenFrameworks를 활용하여 자신만의 게임을 구현해 볼 수 있어 좋았다. 특히 BFS를 사용하여 경로를 탐색하는 알고리즘을 적용하면서, 이론적으로 배운 내용을 실제 프로젝트에 활용할 수 있었다. 시간이 좀 부족해서 더 추가하고 싶었던 기능들 못 써서 좀 아쉬었다.

이 게임에는 컴퓨터가 길을 찾아서 물표물까지 이동하는 것이라 사용자가 그냥 지켜보가만 하면 되는데, 사용자한테 키보드의 키 입력을 받으면서 일반 뱀 게임을 만들어도 좋을 것 같다. 그리고 그 안에서, 테트리스 게임 한 실습시간에 랭킹 기능이랑 비슷하게 이 게임에도 게임을 할 때마다 점수를 기록하여 랭킹을 표시하는 기능을 추가하고 이름을 입력해, 일치하는 점수를 찾아주는 기능도 넣어보면 더 재미있을 것 같다.