C++ 배틀쉽 프로젝트 2-2

변경점

20171701
정지현

BATTLESHIPAPP.CPP - ATTACK()

- 공격에 대한 결과를 리턴하는 Attack 함수를 분리했다.

- flag 변수를 이용해서 공격의 결과를 리턴하게 된다.

```
string BattleShipApp::Attack(char row, char col){
    m_pInputPane -> Draw();
    string flag = "miss";
    std::string s = m_pMap -> attack((int)row - 65, (int)col - 49);
    if (s == "hit"){
        std::string p = m_playerMap -> attack((int)row - 65, (int)col - 49);
        if (p == "hit"){
            flag = "hit";
            char c = m_pMap -> getData((int)row - 65, (int)col - 49);
            char temp;
            switch (c){
                case 'A':
                    if (!aircraft-> isDestroyed()){
                        aircraft -> attacked();
                        m_pStatPane -> turnPass();
                        m_playerMap -> update((int)row - 65, (int)col - 49, 'H');
                        if (aircraft -> isDestroyed()){
                            aircraft -> Draw(m_playerMap -> getWindow());
                            m_pInputPane -> Draw(row, col, "Aircraft Destroyed");
                            flag = "destroyed";
                        }else{
                            m_pInputPane -> Draw(row, col, s);
                        }
                    }
                    break;
```

BATTLESHIPAPP.CPP - GAMEPLAY()

- 처음 원소와 배가 파괴된 이후의 원소는 랜덤으로 찾는다.

```
while (1){
    int temp = rand() % 8;
    int temp_2 = rand() % 8;
    if (check[temp][temp_2] == 0){
        check[temp][temp_2] = 1;
        row = temp + 65;
        col = temp_2 + 49;
        break;
    }
}
```

BATTLESHIPAPP.CPP - GAMEPLAY()

- 스택을 이용한다.
- hit이 된 후 스택에 아래쪽부터 시계 반대 방향으로 원소를 차례로 쌓는다.

```
if (b == "hit"){
    stack<vector<char> > st;
    vector<char> temp1;
    temp1.push_back(row + 1);
    temp1.push_back(col);
    temp1.push_back('S');
    st.push(temp1);
    vector<char> temp2;
    temp2.push_back(row);
    temp2.push_back(col + 1);
    temp2.push_back('E');
    st.push(temp2);
    vector<char> temp3;
    temp3.push_back(row - 1);
    temp3.push_back(col);
    temp3.push_back('N');
    st.push(temp3);
    vector<char> temp4;
    temp4.push_back(row);
    temp4.push_back(col - 1);
    temp4.push_back('W');
    st.push(temp4);
```

BATTLESHIPAPP.CPP - GAMEPLAY()

- 스택에서 원소를 꺼내온다.
- 그 원소를 이용해 공격을 했을때 hit이라는 결과가 나온다면, 진행하던 방향으로 계속 진행하게 된다.
- 그리고 cnt를 하나씩 증가시키게 되는데, 이 부분은 miss를 처리하기 위해서 사용하게 된다.

```
int cnt = 0;
while (!isFinished() && !st.empty()){
    vector<char> vec = st.top();
    st.pop();
    if (vec[0] >= 'A' && vec[0] <= 'H' && vec[1] >= '1' && vec[1] <= '8'){
        string flag = Attack(vec[0], vec[1]);
        usleep(150000);
        check[int(vec[0]) - 65][int(vec[1]) - 49] = 1;
        if (flag == "hit"){
            if (vec[2] == 'N'){
                vector<char> temp;
                temp.push_back(vec[0] - 1);
                temp.push_back(vec[1]);
                temp.push_back('N');
                st.push(temp);
            }
            else if (vec[2] == 'S'){
                vector<char> temp;
                temp.push_back(vec[0] + 1);
                temp.push_back(vec[1]);
                temp.push_back('S');
                st.push(temp);
            }
            else if (vec[2] == 'W'){
                vector<char> temp;
                temp.push_back(vec[0]);
                temp.push_back(vec[1] - 1);
                temp.push_back('W');
                st.push(temp);
```

BATTLESHIPAPP.CPP - GAMEPLAY()

- 배가 파괴된다면 반복을 끝낸다.

```
}else if (flag == "destroyed"){
    cnt = 0;
    break;
```

BATTLESHIPAPP.CPP - GAMEPLAY()

- miss가 나왔을 때 처리하는 부분이다.

- cnt가 0 초과라는 말은 한 칸 이상 진행했다는 말이 되므로 배는 진행하던 방향의 반대 방향에 나머지 부분이 존재하게 된다.

- 그렇기 때문에 반대 방향으로 진행하도록 만들어준다.

```
}else if (flag == "miss" && cnt > 0){
    if (vec[2] == 'N'){
        while (cnt > 0){
            cnt--;
            vec[0]++;
        }
        vector<char> temp;
        temp.push_back(vec[0] + 1);
        temp.push_back(vec[1]);
        temp.push_back('S');
        st.push(temp);
    }else if (vec[2] == 'S'){
        while (cnt > 0){
            cnt--;
            vec[0]--;
        }
        vector<char> temp;
        temp.push_back(vec[0] - 1);
        temp.push_back(vec[1]);
        temp.push_back('N');
        st.push(temp);
```

결론

- 평균 턴 수는 40턴대로 줄었다. 보통 40턴 중~후반으로 나오고 있다.
- 여러 번 실행해 봤을 때 최소 턴 수는 20턴까지 나오지만, 최고 턴 수는 64턴이 나오는 등 편차가 굉장히 컸다.
- 배가 붙어 있을 때의 처리를 하지 못했는데, 그 부분때문에 결과의 편차가 크고 평균 턴수도 그렇게 많이 줄어들지 않은게 아닌가 한다.
- 깃허브 링크: https://github.com/ghyeon0/BattleShip_Project