C++프로그래밍및실습

C++ 미니게임

진척 보고서 #1

모든 기능은 main(임시)에서 확인할 수 있습니다!

제출일자: 2024-11-15
제출자명: 김휘승
제출자학번: 214958

1. 프로젝트 목표 (16 pt)

1) 배경 및 필요성 (14 pt)

수업시간에 terminal 환경에서 수행하는 기초적인 게임들을 배워봤습니다. 흥미로운 게임들이지만, 2인 이상의 플레이어가 필요하거나 볼륨이 작은 게임이 많았습니다. 과제 주제를 고민하던 중, 어렸을 때 폴더폰으로 즐겼던 '미니게임 천국'이 떠올랐고, 이를 참고해 나만의 '미니게임 천국'을 만들면 컴퓨터와 함께 혼자서도 충분히 즐길 수 있는 환경을 제공할 수 있을 것이라 생각했습니다. 또한, 여러 작은 게임을 하나의 프로그램으로 묶음으로써 볼륨 문제도 해결할 수 있다고 판단했습니다. 이와 같은 이유로 나만의 '미니게임 천국'을 만들어 보기로 했습니다.

2) 프로젝트 목표

터미널 환경에서 컴퓨터와 함께 혼자서 즐길 수 있고, 여러가지 게임을 담은 나만의 '미니게임 천국' 만들기

3) 차별점

1. 컴퓨터와의 상호작용: 모든 미니게임에서 컴퓨터가 플레이어의 행동에 반응하며 특정 결과를 제공합니다. 이를 통해 혼자서도 재미있게 즐길 수 있습니다.

2. 점수와 보너스 시스템: 각 미니게임에서 승리할 때마다 점수가 주어지며, 누적된 점수가 일정 기준에 도달하면 다음 게임에서 보너스를 받을 수 있습니다.

3. 랜덤 플레이 모드: 플레이할 게임이 무작위로 결정되는 랜덤 플레이 모드를 통해 게임에 신선함을 더했습니다.

2. 기능 계획

1) 기능 1 : 게임 구현하기

- 설명: 게임은 총 4개 (묵찌빠, 틱택토, 숫자 배열하기, 단어 맞추기)로 구성한다.

(1) 세부 기능 1: 묵찌빠 & 틱택토 게임&컴퓨터 행동 알고리즘

- 설명: 플레이어의 행동 후 다음 행동은 컴퓨터가 수행합니다. 컴퓨터의 행동은 랜덤으로 결정되나, 룰을 준수해야 합니다. (예: 컴퓨터가 돌이 놓여있는 칸에 돌을 놓지 않는다.)

(2) 세부 기능 2: 단어 맞추기 게임& 단어 불러오기

- 설명: 컴퓨터가 제공할 단어는 텍스트 파일에서 불러오고, 텍스트 파일은 단어 주제에 따라 분류합니다. (동물, 과일, 나라이름)

(3) 세부 기능 3 숫자 배열하기 게임& 숫자 배열 알고리즘

- 설명: 컴퓨터가 무작위 숫자를 제공하고, 컴퓨터는 merge sort 알고리즘을 통해 숫자를 크기순으로 정렬합니다. 유저는 특정한 시간 안에 숫자를 크기순으로 정렬해야 합니다.

(4) 세부 기능 4: 누적 점수 기능

- 설명: 게임에서 이기면 점수가 주어지고 누적 점수가 일정 점수를 넘으면 다음 게임에서 보너스를 제공합니다. 묵찌빠는 패배 방어 1회, 틱택토는 추가 행동 1회, 숫자 배열하기는 추가 시간 5초, 단어 맞추기는 질문권을 3회 더 제공합니다.

2) 기능 2 : 메뉴 구현하기

- 설명: 메뉴에서 플레이할 게임을 선택합니다.

(1) 세부 기능 1: 랜덤 플레이

- 설명: 플레이할 게임이 랜덤으로 결정합니다.

(2) 세부 기능 2: 설명 & 점수 표시

- 설명: 메인 메뉴에서 누적 점수를 확인하고, 각 게임의 설명을 제공합니다.

3. 진척사항

1) 게임 구현하기

(1-1) 틱택토 컴퓨터 행동 알고리즘

- **입출력 :** 보드판(map), 좌표(x, y), 보드 한 줄의 칸 수(numCell), 플레이어의 돌(currentPlayer)

- **설명 :** 틱택토 보드판은 vector로 구현, 게임 시작 시 clearMap()함수를 호출, 보드판의 크기를 설정 후, 각 칸을 빈칸(' ')로 초기화한다.

컴퓨터의 행동은 setComputerChoice()함수를 통해 결정된다. 컴퓨터는 보드의 좌표 범위 안의 숫자만 랜덤으로 선택하고, 선택한 수를 isValid()함수를 통해 유효한 좌표를 선택했는지 확인한 후 유효한 좌표면 해당 좌표에 돌을 둔다.

- **적용된 배운 내용 :** vector, 조건문&반복문, 원하는 범위 내의 무작위 수 생성(rand() % numCell, 게임 실행 함수에 srand(time(0))를 추가해 무작위 난수를 생성한다.)

- **코드 스크린샷**

```
void TTT::clearMap(){ // 틱택토의 보드판을 초기화하는 함수
    map = vector<vector<char>>(numCell, vector<char>(numCell)); // 벡터의 1D, 2D 크기를 numCell로 설정하면서 초기화
    for (int i = 0; i < numCell; i++) {
        for (int j = 0; j < numCell; j++) {
            map[i][j] = ' '; // 각 위치를 빈 칸(' ')으로 초기화
        }
    }
}
```

```
void TTT::setComputerChoice(){ // 컴퓨터의 선택을 결정하는 함수
    while(1){ // 유효한 숫자를 결정하기위한 루프
        // 컴퓨터의 입력은 0~numCell-1까지 즉 좌표 범위 안의 숫자만 랜덤으로 생성한다.
        comx = (rand() % numCell);
        comy = (rand() % numCell);
        if (isValid(comx,comy,numCell)==0) { // 유효한 숫자를 선택하면
            map[comx][comy] = currentPlayer; // 틱택토 보드에 컴퓨터의 돌을 둔다.
            break; // 성공적으로 돌을 두면 루프 종료
        }
    }
}
```

```
int TTT::isValid(int x, int y, int numCell){ // 입력한 좌표가 유효한 좌표인지 확인하는 함수
    if (x >= numCell || y >= numCell) { // 좌표 범위를 벗어날때 -100을 반환
        return -100;
    }
    else if (map[x][y] != ' ') { // 입력한 좌표에 돌이 놓여있으면 -200 반환
        return -200;
    }
    else
        return 0; // 유효한 좌표면 0을 반환
}
```

(1-2) 묵찌빠 컴퓨터 행동 알고리즘

- **입출력 :** 사용자의 선택(UserChoice), 컴퓨터의 선택(ComputerChoice), 공격권(turn)

- **설명 :** 1은 묵, 2는 찌, 3은 빠를 나타낸다. 컴퓨터는 1~3까지 무작위의 수를 선택한다.

공격권은 첫번째 가위바위보를 이긴 사람이 가지며 이 상태에서 가위바위보를 비기면 승리, 이기면 공격권 유지, 지면 공격권을 상대에게 넘긴다.

공격권이 설정된 상태에서 두번째 가위바위보를 비기면(사용자의 선택과 컴퓨터의 선택이 같으면) 공격권이 있는 플레이어가 승리한다.

- **적용된 배운 내용 :** 조건문, 원하는 범위 내의 무작위 수 생성(rand() % numCell, 게임 실행 함수에 srand(time(0))를 추가해 무작위 난수를 생성한다.)

- **코드 스크린샷**

```
int MCP::getComputerChoice(){ // 컴퓨터의 선택을 결정하는 함수
    return (rand() % 3 + 1); // 컴퓨터는 1,2,3중 무작위로 하나를 선택한다.
}
```

```
int MCP::turnCheck(int UserChoice, int ComputerChoice){ // 공격권을 선택하는 함수
    // 비기면 0을 반환
    if(ComputerChoice == UserChoice){
        return 0;}
    // 플레이어가 첫번째 가위바위보를 이기면 플레이어가 공격권 획득(100을 반환)
    if((UserChoice == 1&& ComputerChoice == 2) || (UserChoice == 2&& ComputerChoice == 3) || (UserChoice == 3&& ComputerChoice == 1)){
        return 100;
    }
    // 컴퓨터가 첫번째 가위바위보를 이기면 컴퓨터가 공격권 획득(-100을 반환)
    if((ComputerChoice == 1&& UserChoice == 2) || (ComputerChoice == 2&& UserChoice == 3) || (ComputerChoice == 3&& UserChoice == 1)){
        return -100;
    }

    return 0; // 예외값
}
```

```
int MCP::checkWin(int turn, int UserChoice, int ComputerChoice){ // 승자가 결정되었는지 확인하는 함수
    // 유저가 이기면 1000을 반환
    if(turn==100 && (UserChoice == ComputerChoice)){
        return 1000;
    }
    // 컴퓨터가 승리하면 -1000을 반환
    if(turn==-100 && (UserChoice == ComputerChoice)){
        return -1000;
    }
    // 승자가 결정되지 않으면 0을 반환
    return 0;
}
```

(1-3) 게임 구현의 전체적인 특징

- 설명 : 게임 구현이라는 추상적인 내용을 보충 설명하기 위한 내용이다.

- 특징 1 : 모든 게임은 헤더파일(.h)과 실행파일(.cpp)로 구성되어 있고 게임 실행은 메뉴 화면(현재 : main(임시))에서 각 게임의 객체를 생성한 후 객체의 게임실행 메소드를 호출하는 방식으로 이뤄진다.

- 특징 2 : 사용자로부터 입력을 받을 경우 모든 입력은 유효성검사를 거치게 된다. 가령 특정 범위의 숫자(1,2,3)를 입력해야 하는 경우 사용자의 입력한 문자가 숫자인지(isNumber()), 사용자의 입력이 유효한 숫자인지(isValidNumber()) 확인한다.

- 특징 3 : 각 게임 실행파일(.cpp)에서 보너스 객체를 생성한 후 보너스 사용 여부를 물어보고 사용 여부를 isBonusUsed(bool)에 저장 후 보너스 기믹 사용여부를 결정한다.

- 코드 스크린샷

묵찌빠 헤더파일

```
# include <iostream>
#include <string>
using namespace std;

class MCP{
    string UserInput; // 사용자로부터 입력받은 문자열
    int UserChoice; // 유효검사를 완료한 사용자의 선택
    int ComputerChoice; // 컴퓨터의 선택
    int turn; //100이면 사용자의 공격, -100이면 컴퓨터의 공격, 0이면 공격자 미정
    int Winner; // 승자를 표시하는 변수, 사용자가 이기면 1000, 컴퓨터가 이기면 -1000을 반환 0이면 미정
    bool giveScore; // 점수획득 여부를 결정
    bool isBonusUsed; // 보너스가 사용되었는지 확인

    public:
    int getUserInput(); // 사용자로부터 선택을 입력받는 함수
    int getComputerChoice(); // 컴퓨터의 선택을 결정하는 함수
    void showComputerChoice(); // 컴퓨터의 선택을 출력하는 함수
    bool isNumber(string &choice); // 사용자의 입력이 숫자인지 확인하는 함수
    bool isValidNumber(int choice); // 사용자가 입력한 숫자가 유효한 숫자인지 확인하는 함수
    int turnCheck(int UserChoic, int ComputerChoice); // 공격권을 결정하는 함수
    int checkWin(int turn, int UserChoice, int ComputerChoice); // 승자 확인 함수
    int playMCP(); // 실제 게임을 실행시키는 함수
};
```

메인 함수

각 게임의 객체를 생성 후

객체의 게임 실행 함수를 호출한다.

```
#include "Mukchippa.h"
#include "Bonus.h"
#include "tictactoe.h"
#include "Hangman.h"

// 게임을 확인하는 메인(임시)
int main(){
    MCP mcpGame; // 묵찌빠 객체
    Bonus gameBonus; // 보너스 객체
    TTT tictactoeGame; // 틱택토 객체

    mcpGame.playMCP();
    tictactoeGame.playTTT();
```

```cpp
bool MCP::isNumber(string &choice){ // 사용자의 입력이 숫자인지 확인하는 함수
    // 사용자의 문자열의 문자에 대해서
    for (char c : choice) {
        if (!isdigit(c)) return false; // 문자중 숫자가 아닌게 있으면 false를 반환
    }
    return true; // 문자가 전부 숫자면 true를 반환
}

bool MCP::isValidNumber(int choice){ // 사용자의 입력이 유효한 숫자인지 확인하는 함수
    // 사용자가 1,2,3 중 하나를 입력했으면 true를, 아니면 false를 반환
    return choice == 1 || choice == 2 || choice == 3;
}
```

```cpp
bool TTT::isNumber(string a, string b){ // 사용자의 입력이 숫자인지 확인하는 함수
    try { // stoi 예외처리를 위한 try-catch
            // 사용자의 String 입력을 int로 캐스팅한 값을 저장
            x = std::stoi(a);
            y = std::stoi(b);
        }
    catch (const std::invalid_argument& e) { // 숫자가 아닌 값을 입력했을 때
            cout << "입력한 값이 숫자가 아닙니다. 다시 입력하세요." << endl;
            return false; // false를 반환
        }
    catch (const std::out_of_range& e) { // 너무 큰 숫자를 입력했을 때
            cout << "입력한 숫자가 너무 큽니다. 다시 입력하세요." << endl;
            return false; // false를 반환
        }
        return true; // 유효한 숫자면 true를 반환
}

int TTT::isValid(int x, int y, int numCell){ // 입력한 좌표가 유효한 좌표인지 확인하는 함수
    if (x >= numCell || y >= numCell) { // 좌표 범위를 벗어날때 -100을 반환
        return -100;
    }
    else if (map[x][y] != ' ') { // 입력한 좌표에 돌이 놓여있으면 -200 반환
        return -200;
    }
    else
        return 0; // 유효한 좌표면 0을 반환
}
```

보너스 객체 생성 & 보너스 사용 여부 확인

```cpp
int MCP::playMCP(){
    srand(time(0)); // 컴퓨터의 무작위 선택을 위한 난수 생성
    Bonus mcpbonus; // 보너스 시스템을 위한 보너스 객체
    isBonusUsed = mcpbonus.checkUseBonus(); // 보너스 사용여부를 확인

    // 보너스를 사용여부가 true이면 보너스를 사용하는 함수를 호출
    if(isBonusUsed == true){
            mcpbonus.useBonus();
    }
```

(4-1) 누적 점수 기능

- **입출력 :** 누적점수(score), 남은 보너스(bonus), 사용된 보너스(usedbonus)

- **설명 :** Bonus 객체를 사용하는 모든 객체에서 스코어, 보너스, 사용된 보너스를 공유하기 위해 변수를 static으로 설정했으며, 변수의 값을 실수로라도 바꾸는 상황을 줄이기 위해 접근 제어자를 private로 하고 getter, setter로 접근하도록 구성했다.

- **적용된 배운 내용 :** 접근제어자, 캡슐화, 정적변수(static)

- **코드 스크린샷**

```
class Bonus{
    private: // 점수 관련 변수의 접근 제어를 위한 private
    // Bonus 객체를 사용하는 모든 객체에서 스코어, 보너스, 사용한 보너스를 공유하기 위한 static
    // static이 없으면 값이 공유되지 않는다.
    static int score; // 점수
    static int bonus; // 보너스
    static int usedbonus; // 사용된 보너스
    public:
    // 캡슐화한 score, bonus, usedbonus의 getter, setter
    int getScore();
    int getBonus();
    int getUsedBonus();
    void setScore(int record);
    void setBonus(int record);
    void setUsedBonus(int use);
```

```
int Bonus::score = 0;
int Bonus::bonus = 0;
int Bonus::usedbonus = 0;

void Bonus::setScore(int record){ // 점수를 설정
    score = record;
}

int Bonus::getScore(){ // 점수를 반환
    return score;
}

void Bonus::setBonus(int record){ // 남은 보너스를 설정
    bonus = getScore()/200 - getUsedBonus(); // 남은 보너스 = 점수/200 - 사용한 보너스의 수
}

int Bonus::getBonus(){  // 남은 보너스를 반환
    return bonus;
}

void Bonus::setUsedBonus(int use){  // 사용한 보너스의 수를 더하는 함수
    usedbonus += use;   // 매개변수의 값만큼 사용한 보너스의 수를 더함
}

int Bonus::getUsedBonus(){  // 사용한 보너스를 반환
    return usedbonus;
}
```

(4-2) 보너스 사용 기능

- **입출력 :** 변수의 getter, setter

- **설명 :** 사용자로부터 유효한 입력(y,n 대문자 가능)을 받은 후 사용 여부를 결정한다.

보너스가 사용되면 사용된 보너스를 1 올린 후 남은 보너스를 갱신한다. 그 후 각 게임마다 보너스 기믹이 발동된다.

- **적용된 배운 내용 :** 함수화, 캡슐화, 정적변수(static)

- **코드 스크린샷**

```
bool Bonus::checkUseBonus(){    // 보너스 사용여부를 확인하는 함수
    string input; // 사용자의 입력
    while(true){
        if(getBonus() >= 1){ // 남은 보너스가 1 이상이면
            cout << "보너스를 사용하시겠습니까? 보유한 보너스:" << getBonus() << "개" << endl;
            cout << "사용하시려면 'y'를, 사용하지 않으시려면 'n'을 입력하세요."<<endl;
            cin >> input;
            if (input == "y" || input == "Y") { // 사용자의 입력이 y이면 사용여부 true를 반환
                return true;
            }
            else if (input == "n" || input == "N") { // 사용자의 입력이 n이면 사용여부 false를 반환
                return false;
            }
            else { // 사용자의 입력이 y,n이 아니면 루프를 반복하여 입력을 다시 받음
                cout << "잘못된 입력입니다. y 또는 n을 입력해주세요."<<endl;
            }
        }
        else
            return false; // 남은 보너스가 0개이면 false를 반환
    }
}

bool Bonus::useBonus(){ // 보너스를 사용하는 함수
    bool check = checkUseBonus();
    if(check){
        setUsedBonus(1); // 사용된 보너스를 1 증가
        setBonus(getScore()); // 남은 보너스를 갱신}
        return true;
    }
    else
        return false;
}
```

묵찌빠, 틱택톡 객체에서 보너스 객체 생성

```
Bonus TTTbonus; // 보너스 시스템을 위한 보너스 객체
isBonusUsed = TTTbonus.useBonus(); // 보너스 사용여부를 확인
```

```
Bonus mcpbonus; // 보너스 시스템을 위한 보너스 객체
isBonusUsed = mcpbonus.useBonus(); // 보너스 사용여부를 확인
```

묵찌빠, 틱택토의 보너스 기믹 (패배 방어권, 돌 한번 더 두기)

```
// 보너스가 사용되었으면 처음으로 돌아감 (= 아직 turn이 넘어가지 않았으므로 플레이어가 돌을 한번 더 둔다.)
if(isBonusUsed == true){
    isBonusUsed = false; // 보너스가 중복으로 사용됨을 막기위해 false를 대입
    continue;
}
```

```
else if (Winner == -1000) { // 컴퓨터가 승리
    // 보너스 사용여부가 true 이면
    if(isBonusUsed == true){
        cout<<"패배 방어권을 사용합니다. 컴퓨터의 공격을 다시 막으세요."<< endl;
        isBonusUsed = false; // 패배 방어권 재사용을 막기 위해 if문의 조건을 false로 변경
        continue; // 패배하지 않고 다시 방어할 기회를 제공
    }
```

2) 테스트 결과

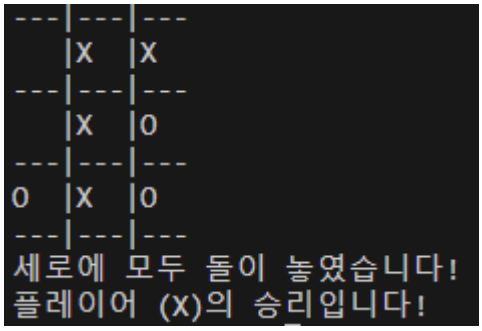
(1-1) 틱택토 컴퓨터 행동 알고리즘

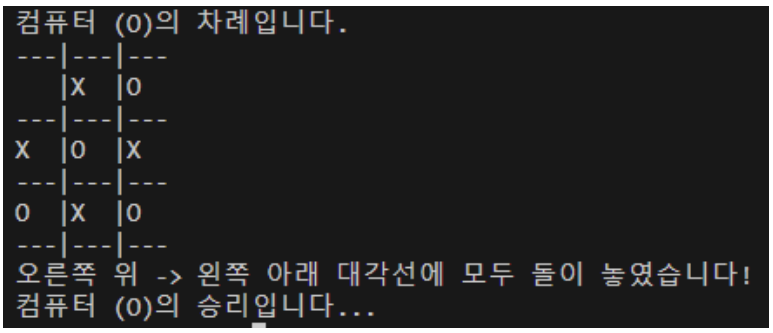
- 설명 : 유저가 돌을 둔 후 컴퓨터도 랜덤으로 돌을 둔다.

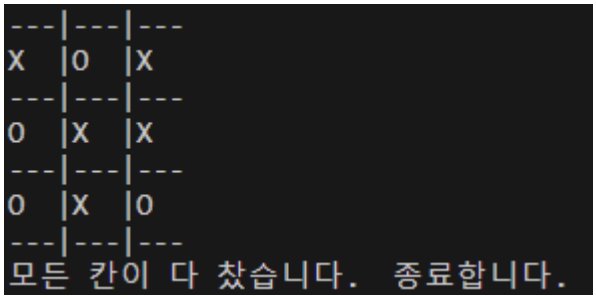
- 테스트 결과 스크린샷

```
유저 (X)의 차례입니다. -> (x, y) 좌표를 입력하세요: 1 1
---|---|---
   |   |
---|---|---
   |X  |
---|---|---
   |   |
---|---|---
```

```
컴퓨터 (O)의 차례입니다.
---|---|---
   |   |
---|---|---
   |X  |
---|---|---
O  |   |
---|---|---
유저 (X)의 차례입니다. -> (x, y) 좌표를 입력하세요:
```







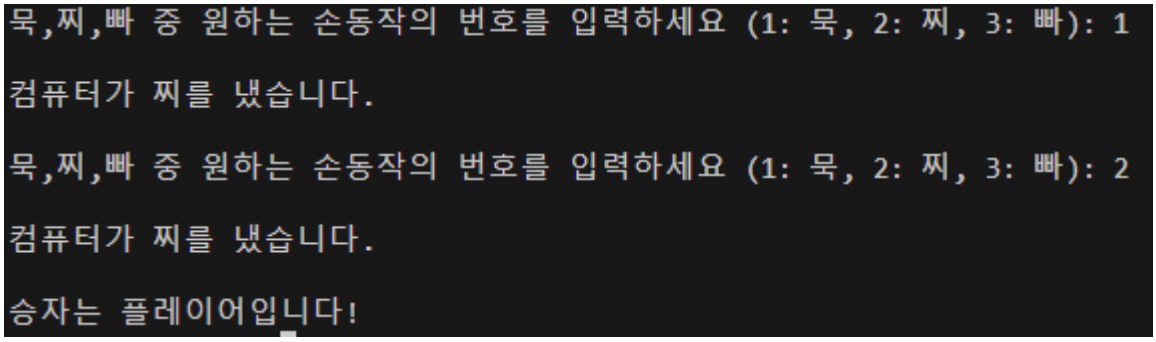
틱택토의 3가지 상황 승리, 패배, 무승부

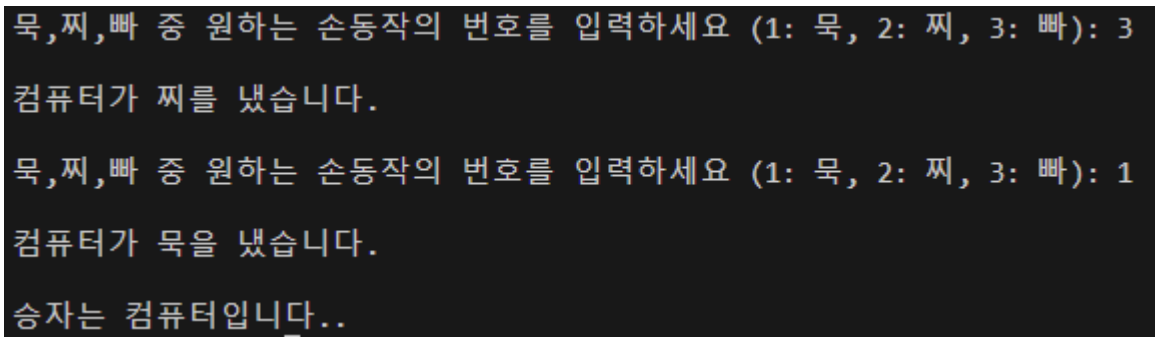
(1-2) 묵찌빠 컴퓨터 행동 알고리즘

- 설명 : 컴퓨터가 무작위로 가위바위보를 한다.

- 테스트 결과 스크린샷

묵찌빠의 2가지 상황 승리, 패배





(4-1), (4-2) 누적 점수기능 & 보너스 사용 기능

- 설명 : 승리 시 점수 획득, 패배 또는 무승부 시 점수 획득 불가, 보너스 사용

- 테스트 결과 스크린샷

점수, 보너스 설정 후 묵찌빠, 틱택토 연속으로 실행

```
// 게임을 확인하는 메인(임시)
int main(){
    MCP mcpGame; // 묵찌빠 객체
    Bonus gameBonus; // 보너스 객체
    TTT tictactoeGame; // 틱택토 객체

    cout<<"초기 점수 : "<<gameBonus.getScore()<<endl;
    cout<<"초기 보너스 : "<<gameBonus.getBonus()<<endl;
    gameBonus.setScore(gameBonus.getScore()+400); // 점수 설정
    gameBonus.setBonus(gameBonus.getScore()); // 남은 보너스 갱신
    cout<<"설정된 점수 : "<<gameBonus.getScore()<<endl;
    cout<<"설정된 보너스 : "<<gameBonus.getBonus()<<endl;

    mcpGame.playMCP(); // 묵찌빠 실행

    tictactoeGame.playTTT(); // 틱택토 실행

    cout<<"최종 점수 : "<<gameBonus.getScore()<<endl;
    cout<<"최종 남은 보너스 : "<<gameBonus.getBonus()<<endl;
    cout<<"사용된 보너스 : "<<gameBonus.getUsedBonus()<<endl;

}
```

```
초기 점수 : 0
초기 보너스 : 0
설정된 점수 : 400
설정된 보너스 : 2
보너스를 사용하시겠습니까? 보유한 보너스:2개
사용하시려면 'y'를, 사용하지 않으시려면 'n'을 입력하세요.
y
```

보너스 사용 여부 확인 (y)

```
묵,찌,빠 중 원하는 손동작의 번호를 입력하세요 (1: 묵, 2: 찌, 3: 빠): 3

컴퓨터가 빠를 냈습니다.

패배 방어권을 사용합니다. 컴퓨터의 공격을 다시 막으세요.
묵,찌,빠 중 원하는 손동작의 번호를 입력하세요 (1: 묵, 2: 찌, 3: 빠):
```

묵찌빠 보너스 기믹 패배 방어권

```
묵,찌,빠 중 원하는 손동작의 번호를 입력하세요 (1: 묵, 2: 찌, 3: 빠): 3

컴퓨터가 빠를 냈습니다.

승자는 플레이어입니다!
보너스를 사용하시겠습니까? 보유한 보너스:1개
사용하시려면 'y'를, 사용하지 않으시려면 'n'을 입력하세요.
y
```

보너스 사용 여부 확인 (y)

```
유저 (X)의 차례입니다. -> (x, y) 좌표를 입력하세요: 0 0
---|---|---
X  |   |
---|---|---
   |   |
---|---|---
   |   |
---|---|---
```

```
유저 (X)의 차례입니다. -> (x, y) 좌표를 입력하세요: 0 1
---|---|---
X  |X  |
---|---|---
   |   |
---|---|---
   |   |
---|---|---
```

틱택토 보너스 기믹 추가 행동권

```
컴퓨터 (O)의 차례입니다.
---|---|---
X  |X  |
---|---|---
   |   |
---|---|---
O  |   |
---|---|---
```

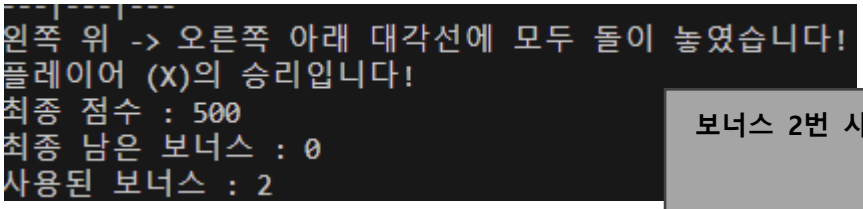
```
X  |X  |O
---|---|---
X  |X  |
---|---|---
O  |X  |O
---|---|---
세로에 모두 돌이 놓였습니다!
플레이어 (X)의 승리입니다!
최종 점수 : 600
최종 남은 보너스 : 1
사용된 보너스 : 2
```

승리시 +100

2승 했으므로 남은 점수는 400+200

200점당 보너스 = 남은 보너스 1개

사용된 보너스 = 2개



보너스 2번 사용, 묵찌빠 패배, 틱택토 승리

승리시 +100

1승 했으므로 남은 점수는 400+100

200점당 보너스 = 남은 보너스 0개

사용된 보너스 = 2개

5. 프로젝트 일정

(진행한 작업과 진행 중인 작업 등을 표기)

업무		11/3	11/10	11/17	11/24	12/1	12/8
제안서 작성		완료					
기능1	세부기능1		완료				
	세부기능2			진행중			
	세부기능3				------->		
	세부기능4		진행중				
기능2	세부기능1					------->	
	세부기능2					---------->	