🎮 100을 넘어라! 게임

응용 7_2_2 업그레이드

2020675058 전유빈

게임 개요

기본 규칙

플레이어와 컴퓨터가 번갈아 1~10 사이의 숫자를 입력하고, 누적 합이 100 이상이 되면 게임이 종료됩니다.

🎯 승리 조건

100을 넘기는 순간, 상대방이 승리합니다!

Easy Mode

컴퓨터가 랜덤으로 선택 (기존 게임)

Hard Mode

전략 기반 컴퓨터와 게임

```
// Hard Mode: 전략적 선택
if (difficulty == 2 && lastPlayerInput > 0) {
    input = 11 - lastPlayerInput;

    // 컴퓨터가 이미 사용한 숫자면 랜덤으로
    for (int i = 0; i < computerCount; i++) {
        if (computerUsed[i] == input) {
            input = -1;
            break;
        }
    }
}

// Easy Mode 또는 전략 실패 시: 랜덤 선택
if (difficulty == 1 || input == -1) {
    do {
        valid = 1;
        input = (rand() % MAX_INPUT) + 1;

        // 컴퓨터 본인의 중복 체크만
        for (int i = 0; i < computerCount; i++) {
            if (computerUsed[i] == input) {
                valid = 0;
                break;
            }
        }
    } while (!valid);
}
```

게임 모드 비교

Easy Mode

랜덤 선택 방식

컴퓨터는 1~10 사이의 숫자를 무작위로 선택합니다.

```
rand() % 10 + 1
```

초보자에게 적합한 난이도로, 운에 따라 승패가 결정됩니다.

Hard Mode

전략적 계산 방식

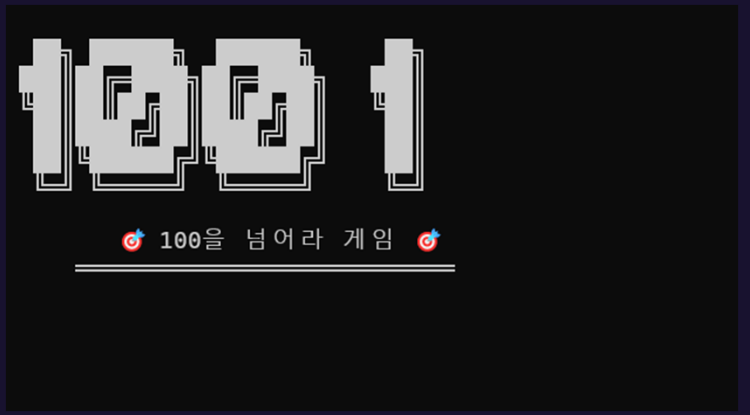
컴퓨터는 11 - 플레이어 입력 공식을 사용하여 승리를 방지합니다.

```
input = 11 - lastPlayerInput
```

AI가 상대방의 승리를 차단하는 전략적 선택으로 높은 난이도를 제공합니다.

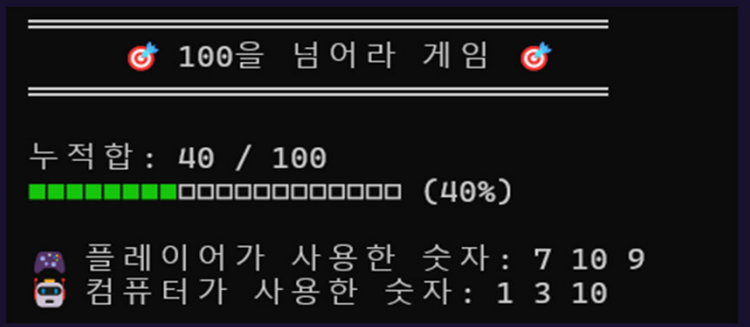
UI 개선 및 시각적 요소

✨ 인트로 화면



게임 시작 시 화려한 ASCII 아트 타이틀과 점멸 효과로 플레이어를 맞이합니다.

색상 변화 효과: 타이틀 텍스트는 특정 주기로 색상이 변하며,
이는 시선을 사로잡는 동적인 요소를 제공하여 게임의 시작을 더욱 인상 깊게 만듭니다.



📊 진행 바 시각화

게임의 진행 상황을 직관적으로 보여주는 동적인 진행 바(Progress Bar)를 도입합니다.
이는 플레이어가 현재 게임 진행상황을 직관적으로 파악하고 전략을 정비하도록 돕습니다.

현재 진행도

게임 단계, 로딩 상태 등 다양한 진행 상황을
막대 형태로 시각화합니다.

직관적인 피드백

남은 시간이나 완료까지의 퍼센티지를 숫자로 함께 표시하여
사용자에게 정확한 정보를 제공합니다.

사운드 효과 추가

게임의 몰입감을 극대화하기 위해 다양한 사운드 효과를 추가했습니다.

```
// 사운드 출력
void playSound(int freq, int duration) {
    Beep(freq, duration);
}
```

게임 시작 BGM

경쾌한 시작 멜로디를 재생합니다.

```
// 시작 사운드
playSound(523, 150);  // C
playSound(659, 150);  // E
playSound(784, 200);  // G
```

승리 사운드

승리의 기쁨을 표현하는 경쾌한 멜로디를 출력합니다.

```
// 승리 사운드
playSound(523, 150);
playSound(659, 150);
playSound(784, 150);
playSound(1047, 300);
```

패배 사운드

아쉬움을 나타내는 하강 음계를 재생합니다.

```
// 패배 사운드
playSound(392, 200);
playSound(330, 200);
playSound(262, 400);
```

게임 흐름 및 기술 구조

게임 시작
모드 선택 및 초기화

플레이어 입력
getch()로 입력 받기

컴퓨터 입력
rand() 또는 전략 계산

게임 종료
승자 결정 및 기록

핵심 기술

- **입력 관리**: getch()로 실시간 사용자 입력 처리
- **랜덤 생성**: rand()로 Easy Mode 구현
- **전략 AI**: 11 - lastPlayerInput 공식으로 Hard Mode 구현
- **상태 관리**: 누적합 추적 및 게임 종료 조건 체크

```
if (difficulty == 1 || input == -1) {
    do {
        valid = 1;
        input = (rand() % MAX_INPUT) + 1;

        // 컴퓨터 본인의 중복 체크만
        for (int i = 0; i < computerCount; i++) {
            if (computerUsed[i] == input) {
                valid = 0;
                break;
            }
        }
    } while (!valid);
}
```

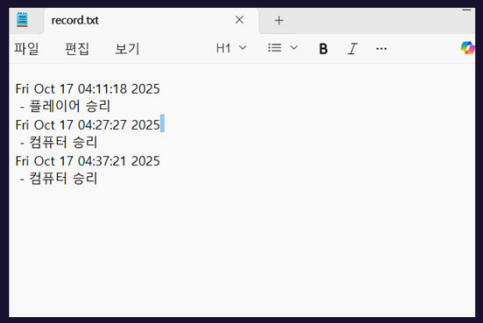
```
// ========= 진행 막대 그리기 =========
void drawProgressBar(int current, int max) {
    int filled = (current * BAR_LENGTH) / max;

    printf("  누적합: %d / %d\n  ", current, max);

    setColor(COLOR_SUCCESS);
    for (int i = 0; i < filled; i++) {
        printf("■");
    }
    setColor(COLOR_DEFAULT);
    for (int i = filled; i < BAR_LENGTH; i++) {
        printf("□");
    }
    printf(" (%d%%)\n", (current * 100) / max);
```

```
// Hard Mode: 전략적 선택
if (difficulty == 2 && lastPlayerInput > 0) {
    input = 11 - lastPlayerInput;

    // 컴퓨터가 이미 사용한 숫자면 랜덤으로
    for (int i = 0; i < computerCount; i++) {
        if (computerUsed[i] == input) {
            input = -1;
            break;
        }
    }
}
```

파일 저장 및 기록 관리

1. **게임 진행**
   플레이어와 컴퓨터의 대결
2. **결과 계산**
   승패 판정 및 데이터 수집
3. **파일 저장**
   `fopen()`, `fprintf()` 사용
4. **기록 확인**
   record.txt에서 이력 조회

```c
// ========= 기록 저장 =========
void saveRecord(int isPlayer) {
    FILE* fp = fopen("record.txt", "a");
    if (fp != NULL) {
        time_t now = time(NULL);
        fprintf(fp, "%s - %s\n", ctime(&now), isPlayer ? "플레이어 승리" : "컴퓨터 승리");
        fclose(fp);
    }
}
```



record.txt 파일

게임의 승리/패배 기록을 자동으로 저장하여, 게임이 끝난 후 이전 게임 기록을 참고할 수 있습니다. 통계 분석과 실력 향상 추적이 가능합니다.

통계 제공

승률과 게임 패턴 분석 가능

반복 학습

이전 게임에서 배우고 개선

목표 설정

기록 경신을 위한 동기 부여

코드 구조 및 모듈화

체계적인 함수 분리로 가독성과 유지보수성을 향상시켰습니다.

intro()

게임 인트로 화면 및 타이틀 표시

showWinner()

승리/패배 결과 처리 및 출력

drawProgressBar()

현재 누적합의 진행 상태 시각화

getComputerInput()

AI의 전략적 다음 수 계산

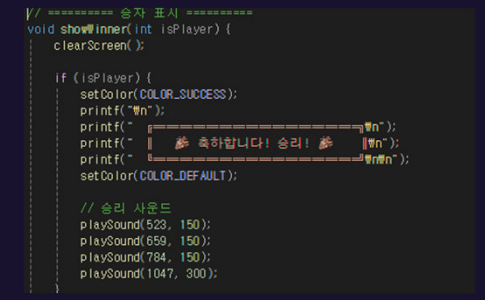
gameLoop()

메인 게임 루프 및 턴 관리

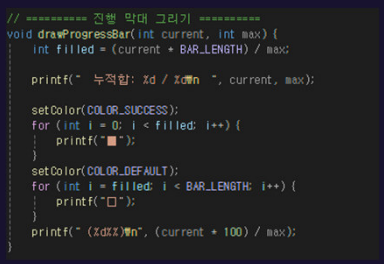
saveRecord()

게임 결과를 파일에 기록

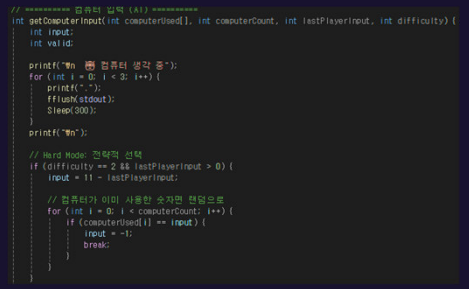
```
// ========== 인트로 화면 ==========
void intro() {
    clearScreen();
    setColor(COLOR_TITLE);

    // ASCII 아트 타이틀 (점멸 효과)
    for (int blink = 0; blink < 3; blink++) {
        clearScreen();
        printf("\n");
        printf("  ██╗ ██████╗  ██████╗     ██████╗ \n");
        printf("  ██║ ██╔═══██╗ ╚═══██╗    ╚═══██╗\n");
        printf("  ██║ ██║   ██║  ████╔╝     ████╔╝\n");
        printf("  ██║ ██║   ██║ ██╔═══╝     ╚═══██╗\n");
        printf("  ██║ ╚██████╔╝ ███████╗    ██████╔╝\n");
        printf("  ╚═╝  ╚═════╝  ╚══════╝    ╚═════╝ \n");
        printf("\n");
        setColor(COLOR_DEFAULT);
        printf("          🎯 100을 넘어라 게임 🎯\n");
        printf("     ═══════════════════════\n\n");
```



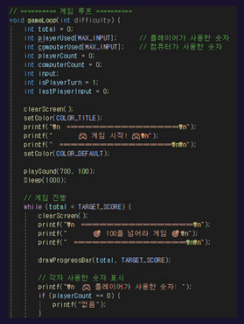
```
// ========== 승자 표시 ==========
void showWinner(int isPlayer) {
    clearScreen();

    if (isPlayer) {
        setColor(COLOR_SUCCESS);
        printf("\n");
        printf("  ╔═══════════════════════╗\n");
        printf("  ║   🎉 축하합니다! 승리! 🎉   ║\n");
        printf("  ╚═══════════════════════╝\n\n");
        setColor(COLOR_DEFAULT);

        // 승리 사운드
        playSound(523, 150);
        playSound(659, 150);
        playSound(784, 150);
        playSound(1047, 300);
    }
```



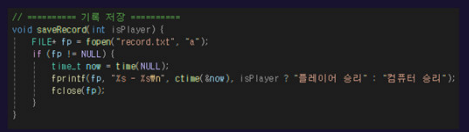
```
// ========== 진행 막대 그리기 ==========
void drawProgressBar(int current, int max) {
    int filled = (current * BAR_LENGTH) / max;

    printf("  누적합: %d / %d\n  ", current, max);

    setColor(COLOR_SUCCESS);
    for (int i = 0; i < filled; i++) {
        printf("■");
    }
    setColor(COLOR_DEFAULT);
    for (int i = filled; i < BAR_LENGTH; i++) {
        printf("□");
    }
    printf(" (%d%%)\n", (current * 100) / max);
}
```



```
// ========== 컴퓨터 입력 (AI) ==========
int getComputerInput(int computerUsed[], int computerCount, int lastPlayerInput, int difficulty) {
    int input;
    int valid;

    printf("\n  🤖 컴퓨터 생각 중");
    for (int i = 0; i < 3; i++) {
        printf(".");
        fflush(stdout);
        Sleep(300);
    }
    printf("\n");

    // Hard Mode: 전략적 선택
    if (difficulty == 2 && lastPlayerInput > 0) {
        input = 11 - lastPlayerInput;

        // 컴퓨터가 이미 사용한 숫자면 랜덤으로
        for (int i = 0; i < computerCount; i++) {
            if (computerUsed[i] == input) {
                input = -1;
                break;
            }
        }
    }
```



```
// ========== 게임 루프 ==========
void gameLoop(int difficulty) {
    int total = 0;
    int playerUsed[MAX_INPUT];    // 플레이어가 사용한 숫자
    int computerUsed[MAX_INPUT];  // 컴퓨터가 사용한 숫자
    int playerCount = 0;
    int computerCount = 0;
    int input;
    int isPlayerTurn = 1;
    int lastPlayerInput = 0;

    clearScreen();
    setColor(COLOR_TITLE);
    printf("\n  ═══════════════════════\n");
    printf("          🎮 게임 시작! 🎮\n");
    printf("  ═══════════════════════\n\n");
    setColor(COLOR_DEFAULT);

    playSound(700, 100);
    Sleep(1000);

    // 게임 진행
    while (total < TARGET_SCORE) {
        clearScreen();
        printf("\n  ═══════════════════════\n");
        printf("      🎯 100을 넘어라 게임 🎯\n");
        printf("  ═══════════════════════\n\n");

        drawProgressBar(total, TARGET_SCORE);

        // 각자 사용한 숫자 표시
        printf("\n  🧑 플레이어가 사용한 숫자: ");
        if (playerCount == 0) {
            printf("없음");
        }
```



```
// ========== 기록 저장 ==========
void saveRecord(int isPlayer) {
    FILE* fp = fopen("record.txt", "a");
    if (fp != NULL) {
        time_t now = time(NULL);
        fprintf(fp, "%s - %s\n", ctime(&now), isPlayer ? "플레이어 승리" : "컴퓨터 승리");
        fclose(fp);
    }
}
```