Journey To Happiness

Simple Runner Game with C++ and API

Name : 김나영

Tel : 010-3902-6074

e-mail : nayeong0121@gmail.com

목차

1. 게임 개요

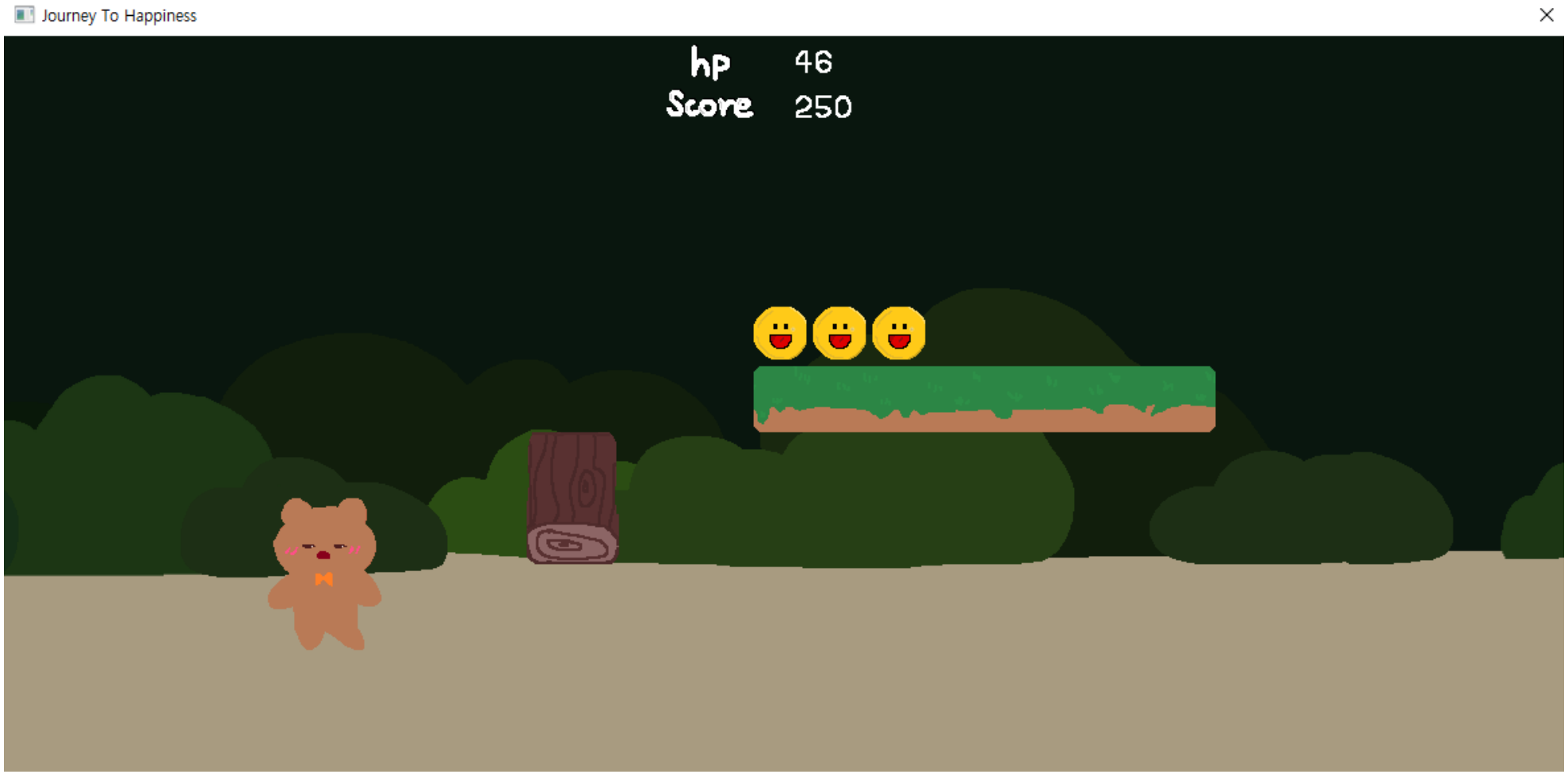
- 언어 : C++
- API : WinAPI
- 플랫폼 : Windows
- 게임 영상 링크
- 깃 허브 링크
- 게임 개발 기간 : 2024.10 ~ 2024.11

- Journey To Happiness는 C++과 WinAPI를 사용하여 개발한 간단한 러너 게임입니다.
- 이 게임은 곰 캐릭터를 조작해 장애물을 피하고 점수를 획득하는 것이 목표입니다.

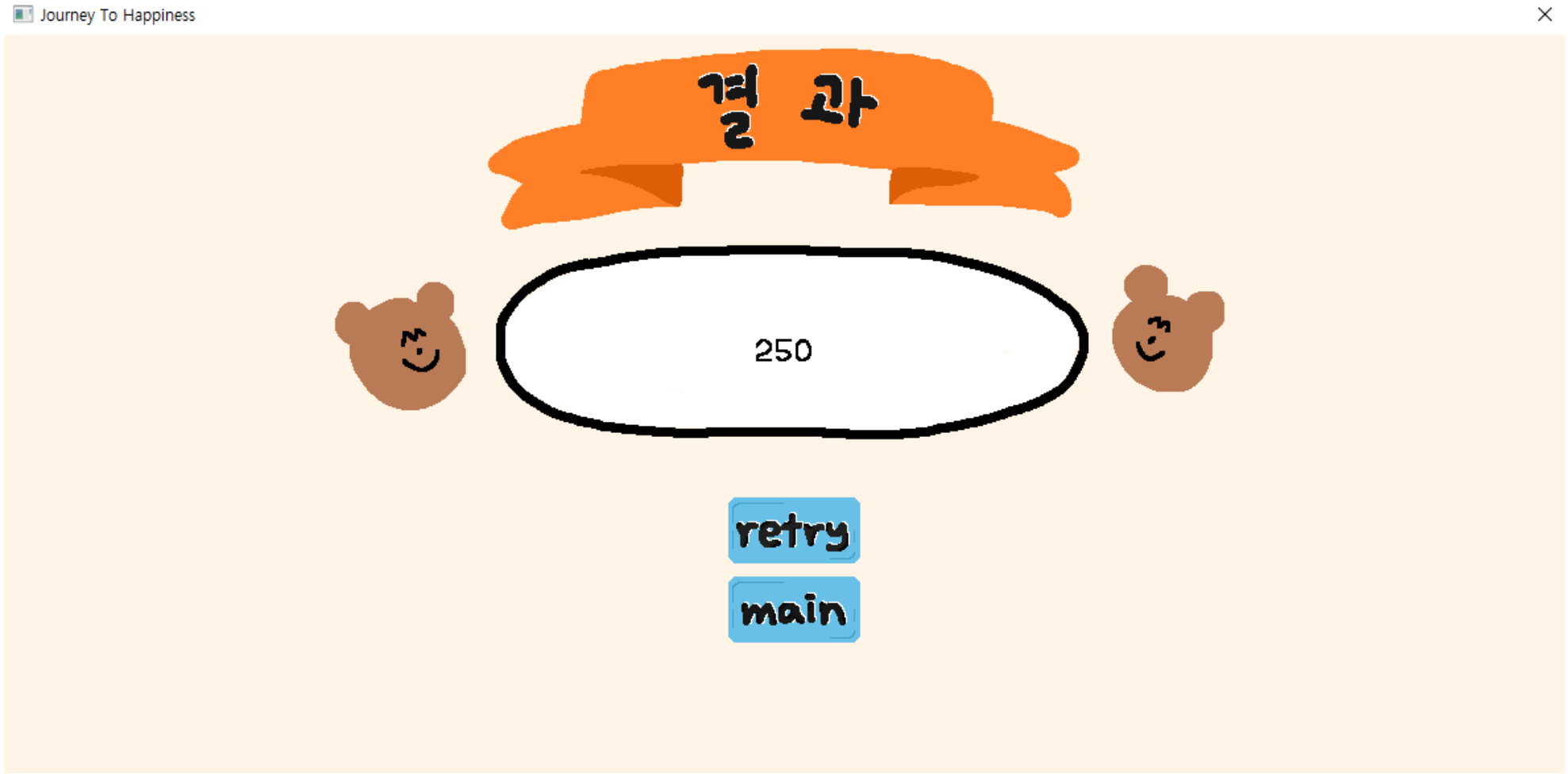
2. 게임 화면 – 메인 화면

2. 게임 화면 – 스테이지 선택 화면

2. 게임 화면 – 게임 플레이 화면



2. 게임 화면 – 게임 종료 화면



2. 게임 화면 – 랭크 화면

3. 기능 구현 및 함수 구현

캐릭터 조작

곰은 슬라이드와 점프 동작이 가능합니다.

점프 후 공중 땅에 착지하거나, 공중 땅이 없으면 자연스럽게 낙하합니다.

게임 환경

배경은 오른쪽에서 왼쪽으로 스크롤되어 곰이 이동하는 느낌을 줍니다.

장애물과 공중 땅도 오른쪽에서 등장하게 되는데, 랜덤으로 나오게 됩니다.

3. 기능 구현 및 함수 구현

```
//곰이 점프하는 함수
void Jump(HWND hWnd, int randNum, OBJ* airGnd, OBJ_MON* objBear) {
    if (!objBear->isJumping) return;    // 점프를 하지 않으면 함수 종료
    static float posY = -1;             // 곰의 top 값 변수

    // 초기 점프 높이를 설정
    if (posY == -1)
        posY = (float)objBear->rect.top;

    // 점프 시간이 착지 시간을 넘으면 착지 처리
    if (jumpTime >= landTime) {
        jumpTime = 0.0f;                // 점프 시간 초기화
        objBear->isJumping = FALSE;
        objBear->isOnGround = TRUE;
        objBear->rect.top = (long)posY; // 원래 높이로 돌아감
    }
    // 점프 중이라면 계속해서 높이를 계산
    else {
        objBear->rect.top = (long)(posY + (jumpTime * jumpTime - jumpTime * landTime) * 400);
        jumpTime += 0.04f;
        objBear->isOnGround = FALSE;    // 점프 중이므로 착지 상태 해제
    }
```

점프 기능 함수

3. 기능 구현 및 함수 구현

```
// 곰이 슬라이딩 여부의 좌표를 설정하는 함수
void IsSliding(OBJ_MON* objBear, OBJ* airGnd) {
    //곰이 슬라이딩 중이라면 곰의 슬라이드 표면으로 좌표값 설정
    int bearHeight = objBear->isSliding ? objBear->g_slideBear.nHeight : objBear->g_sfBear->nHeight;
    int bearWidth = objBear->isSliding ? objBear->g_slideBear.nWidth : objBear->g_sfBear->nWidth;

    if (objBear->isSliding) {   //곰이 슬라이딩 중이면
        if (IsOnAirGround((float)objBear->rect.left,    //곰이 공중 땅 위에 있다면
            (float)objBear->rect.top, airGnd, objBear)) //곰의 사각형 top값을 공중 땅 위로 설정
            objBear->rect.top = airGnd->rect.top - bearHeight;
        else                                            //곰이 공중 땅 위에 없다면
            objBear->rect.top = BEAR_SLIDING_ONGROUND;                //곰의 top 값을 일반 땅 위로 설정
    }
    else if (!objBear->isSliding && !objBear->isJumping) {  //곰이 슬라이딩하지 않으면서 점프도 하지 않으면
        if (!IsOnAirGround((float)objBear->rect.left,       //곰이 공중 땅 위에 없다면
            (float)objBear->rect.top, airGnd, objBear)) {
            //곰 낙하
            if (objBear->rect.top < BEAR_IDLE_ONGROUND) {
                objBear->rect.top += 20;
            }
            else
                objBear->rect.top = BEAR_IDLE_ONGROUND;
        }
        else                                                //곰이 공중 땅 위에 있다면
            objBear->rect.top = airGnd->rect.top - bearHeight;  //곰의 top 값을 공중 땅 위로 설정

    }
    objBear->rect.right = objBear->rect.left + bearWidth; //곰의 right 값 설정
    objBear->rect.bottom = objBear->rect.top + bearHeight;  //곰의 bottom 값 설정
}
```

슬라이딩 기능 함수

3. 기능 구현 및 함수 구현

```
//곰이 공중 땅 위에 있는지 판단하여 반환하고 좌표를 설정하는 함수
BOOL IsOnAirGround(float bearX, float bearY, OBJ* airGnd, OBJ_MON* objBear)
{
    //곰이 슬라이딩 하는지 여부에 따라 곰의 그림 높이 설정
    int bearHeight = objBear->isSliding ? objBear->g_slideBear.nHeight : objBear->g_sfBear->nHeight;

    if (objBear->isSliding && objBear->rect.top == BEAR_SLIDING_ONAIRGND)   // 곰이 슬라이딩 중이고 곰이 공중 땅 위에 있다면
        bearY = (float)(airGnd->rect.top - objBear->g_slideBear.nHeight); //공중 땅에서 슬라이드하는 좌표로 변경
    if (!objBear->isSliding && objBear->rect.top == BEAR_IDLE_ONAIRGND)// 곰이 슬라이딩 중이 아니고 곰이 공중 땅 위에 있다면
        bearY = (float)(airGnd->rect.top - objBear->g_sfBear->nHeight); //공중 땅 위에 있는 좌표로 변경

    //공중 땅 갯수만큼 반복
    for (int i = 0; i < 5; i++)
    {
        if (airGnd[i].g_sfObj.isActive) { //공중 땅이 활성화 되어 있다면
            // 공중 땅 범위 내에 곰이 위치한다면
            if (objBear->rect.right >= airGnd[i].rect.left && bearX <= airGnd[i].rect.right &&
                abs((bearY + bearHeight) - airGnd[i].rect.top) <= 5) {
                objBear->rect.top = airGnd[i].rect.top - bearHeight; //곰의 top값을 공중 땅위에 있을 수 있도록 설정
                objBear->isOnGround = TRUE; //곰의 isOnGround를 TRUE로 설정
                return TRUE; //곰이 공중 땅 위에 있으므로 TRUE 리턴
            }
        }
    }
    return FALSE;   //곰이 공중 땅 위에 없으므로 FALSE 리턴
}
```

공중 땅 착지 함수

3. 기능 구현 및 함수 구현

아이템 시스템

장애물 : 충돌 시 체력 감소

스마일 젤리 : 충돌 시 50점 획득

생명 물약 : 충돌 시 체력 5 회복

속도 증가

시간이 지날 수록 게임 속도가 점차 빨라지지만, 최대 속도 제한을 설정했습니다.

3. 기능 구현 및 함수 구현

게임 종료

체력이 0이 되면 게임 종료

최종 점수를 화면에 표시하고, 파일로 저장합니다.

랭킹 시스템

상위 5명의 점수를 랭킹 화면에서 확인 가능합니다.

스테이지 선택

메인 화면에서 게임 시작 버튼을 누르면, 다양한 배경 이미지로 구성된 스테이지를 선택할 수 있습니다.

3. 기능 구현 및 함수 구현

```
// 곰이 장애물과 충돌했는지 확인하는 함수
void MonCollideWithObs(OBJ_MON* objBear, OBJ* objObstacle) {
    RECT temp;
    int bearLife = objBear->nLife;  //bearLife에 곰의 생명력 저장

    //곰과 장애물이 충돌한다면
    if (IntersectRect(&temp, &(objBear->rect), &(objObstacle->rect))) {
        SetObjHP(objBear, --bearLife); //곰의 체력을 감소시킨다.
    }
}
```

캐릭터와 장애물 충돌 여부를 확인하는 함수

3. 기능 구현 및 함수 구현

```
//ScoreData를 파일로 저장하는 함수
void SaveScoreFile(SCORE* scoreData) {
    FILE* out;

    fopen_s(&out, "scoreData.bin", "wb");   //scoreData.bin이라는 이름으로 파일을 쓰는 걸로 파일을 연다.
    if (NULL == out) {
        ::OutputDebugString("출력 파일 개방 실패");
        return;
    }
    Node* curNode = scoreData->scoreList;   //scoreData의 scoreList를 현재 노드로 설정
    int scores[5] = { 0 };                  //scores 배열을 0으로 초기화
    int index = 0;

    //scoreList의 점수들을 scores배열에 넣는다.
    while (curNode != NULL && index < 5) {  //현재 노드가 NULL이 아니고 index가 5보다 작으면 반복
        scores[index++] = curNode->score;   //scores배열에 현재 노드의 점수를 넣고 index 증가
        curNode = curNode->next;            //다음 노드로 현재 노드 변경
    }

    fwrite(scores, sizeof(int), 5, out);    //scores 배열을 파일에 쓴다.
    fclose(out);
}
```

점수를 파일에 저장하는 함수

3. 기능 구현 및 함수 구현

```
// 저장한 점수 리스트 불러오기
int ImportScoreFile(SCORE* scoreData) {
    FILE* in;

    fopen_s(&in, "scoreData.bin", "rb");
    if (NULL == in) {                              // 저장된 파일이 없다면
        ::OutputDebugString("저장된 파일이 없음");
        return 0;
    }

    int scores[5] = { 0 };                //scores 5개 배열 0으로 초기화

    fread(scores, sizeof(int), 5, in);  // scores 배열에 저장한 점수 불러오기

    for (int i = 0; i < 5; i++) {
        if (scores[i] != 0) {             // scores 배열에 저장한 점수가 0이 아니라면
            InsertScore(scoreData, scores[i]);  // scoreData에 불러온 점수 넣기
        }
    }
    fclose(in);
    return 1;
}
```

점수 리스트가 저장된 파일을 불러오는 함수

감사합니다.