**광운 주크 박스 최종보고서**

**팀명 : 언더더C**

2020204019 권혜진

20202040 전성훈

20202040 김효민

목차

1. **프로젝트 구현 및 검증3**

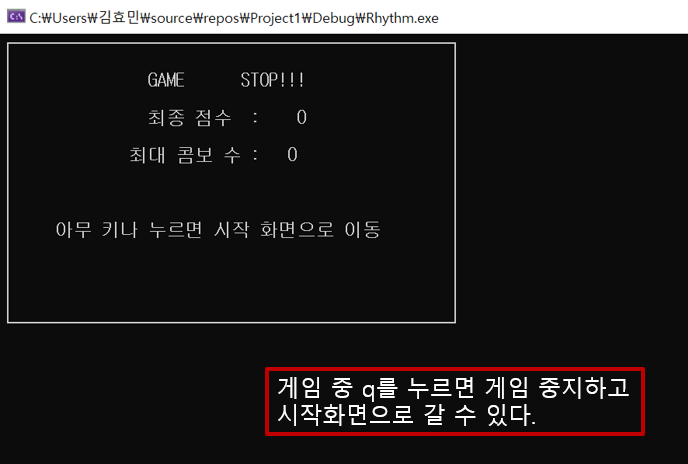
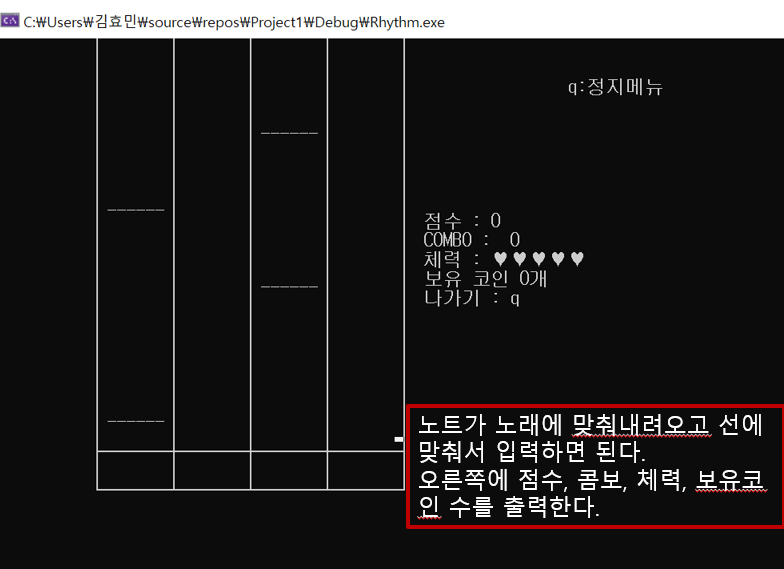
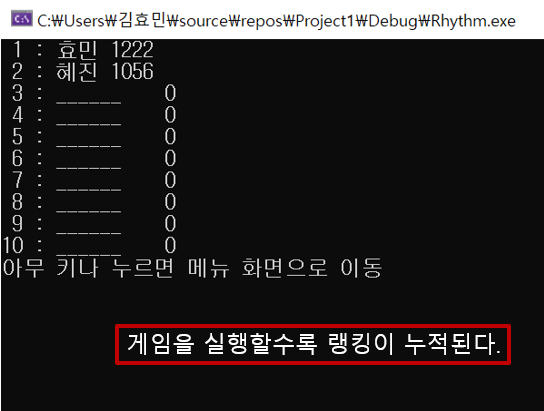
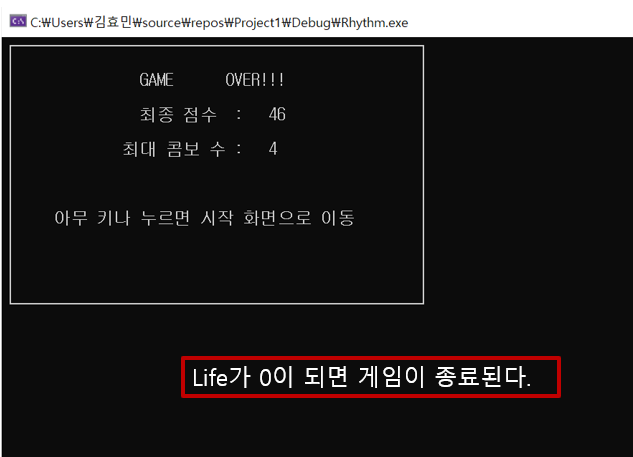
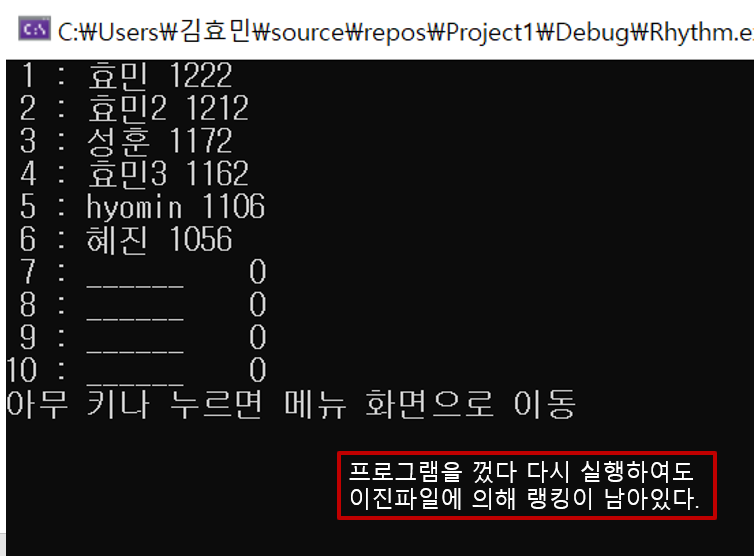
구현 단계

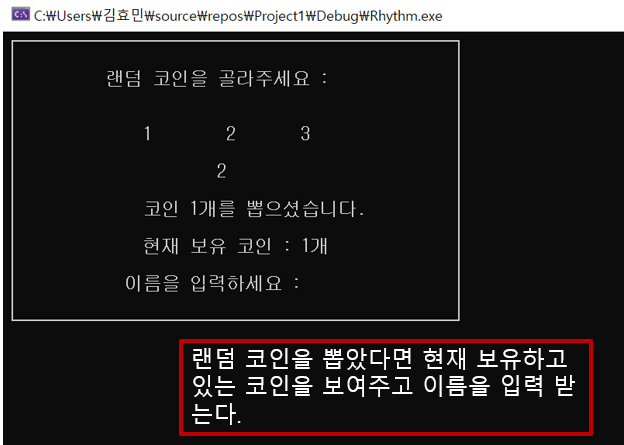
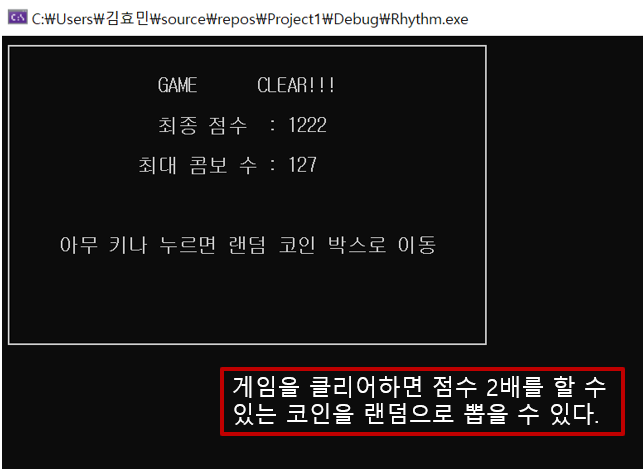
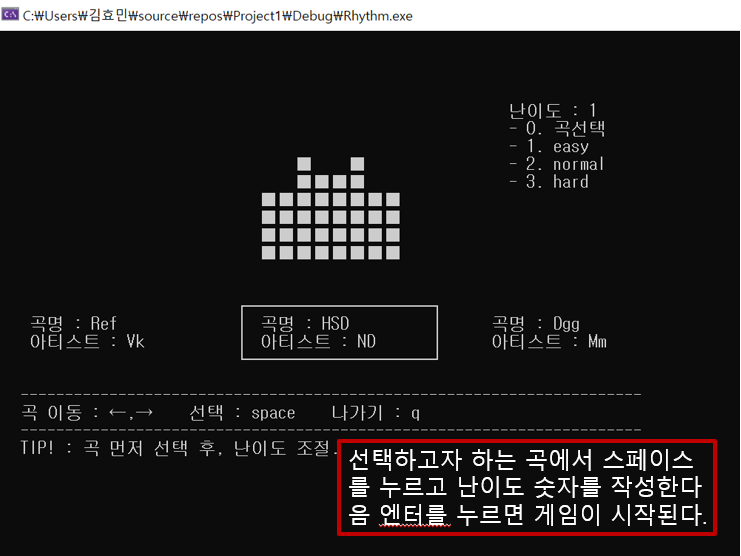
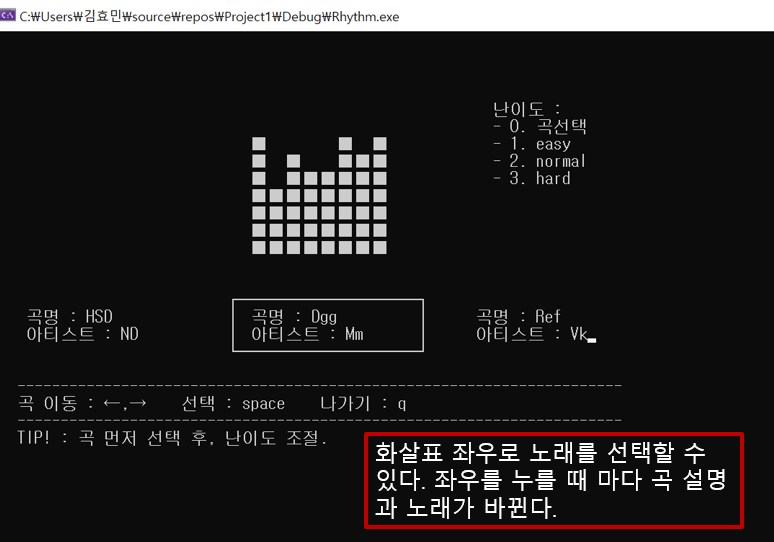
검증 단계

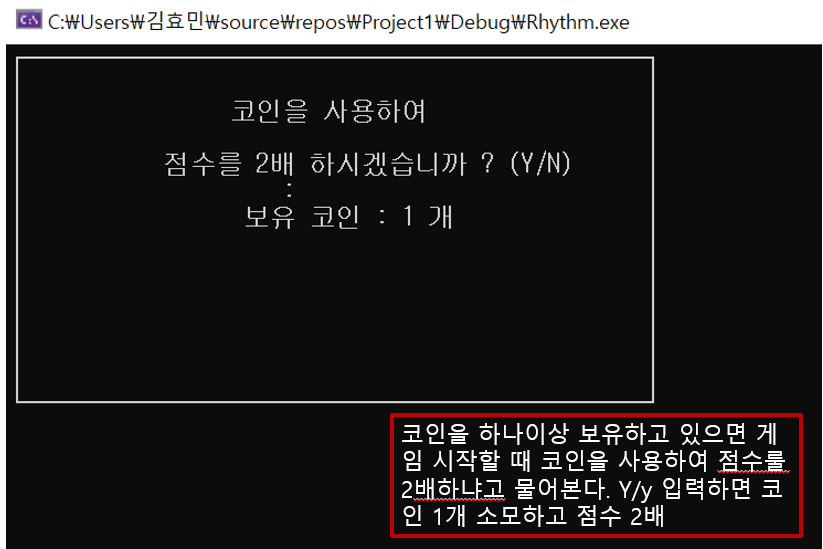
테스트 계획서

1. **프로젝트 진행 과정에 대한 고찰5**

1. **프로젝트 역할 분배7**
2. **프로젝트 구현 및 검증**

**구현단계**

****

****

**검증단계**

* **테스트 개요**
* **명칭 :** 광운 주크박스 – 리듬게임
* **총 기간 :**

프로젝트 구현 - 11월2일~11월25일

테스트 – 11월 24일~11월28일

* **목표 :** 배열을 이용해 노트가 내려오는 기능을 이용하여 리듬게임 구현
* **목적 :** 구현한 리듬게임의 완성도 평가를 위한 최종 기능 및 비기능 요소의 테스트를 수행
* **테스트 종류**

직접 진행한 테스트 목록

1. 단위 시험 테스트 : 개발자가 개별적으로 SW부분 기능 검증
2. txt파일에 따라 배열화가 잘 이루어지고 노트가 인게임에서 출력이 되는지 확인
3. 게임 인터페이스에서 페이지 별로 이동이 잘 이루어지는 지 확인
4. 랭킹 출력되는 지에 대한 확인
5. 노래가 재생되는 지에 대한 확인
6. 애니메이션이 잘 출력이 되는 지에 대한 확인
7. 통합시험 테스트 : 시스템 담당자 및 설계자가 컴포넌트 간 상호작용 테스트
8. 노트가 내려오는 싱크가 현재 출력되고 있는 노래와 알맞은 지에 대한 확인

* 팀원들의 각 컴퓨터마다 성능이 달라 미세한 싱크 차이가 발생함을 알 수 있었음.

1. 노래 선택 화면에서 화살표의 움직임에 따라 노래 선택이 반영이 원활히 이루어지는 지에 대한 확인
2. 게임 진행 과정 중 점수, 라이프, 콤보가 게임 상황에 맞게 적절히 변하는 지에 대한 확인
3. 랭킹 입력이 잘 되는지에 대한 확인
4. 시작화면에서 화살표의 입력에 따라 정상적으로 커서가 움직이는 지에 대한 확인
5. 노트의 정확성(perfect,great…)에 대한 판정이 인게임에서 노트의 y값에 따라 알맞게 판정하는 지에 대한 확인

구상한 테스트 목록

1. 시스템 시험 테스트 : 전문테스트 조직이 시스템 안정성, 속도, 정확성 검증
2. 인게임에서 동시에 키를 입력하였을 때의 메모리 사용량 및 CPU 사용률, 및 응답시간 체크

만족기준 : 0.5초 이내

1. 인수 테스트 : 고객이 시스템 요구사항 만족 정도를 테스트
2. 인수기준 : 노래의 재생이 끝나는 시각과 노트가 전부 내려오는 시각의 차이가 3초 미만

총 시행횟수 : 100회

1. **프로젝트 진행 과정에 대한 고찰**
2. 계획서에서 구상한 이어하기 기능을 구현하지 못했다. 노래 파일 자체는 일시정지하고 다시 재생할 수 있지만, 노트를 그리는 경우에는 일시정지하고 진행할 경우, 현재 진행되고 있던 배열의 노트 값들은 모두 날라가고 그러면 필연적으로 싱크에 오차가 발생한다. 그래서 이어하기 기능을 구현하지 못하고 점수 두배로 대체했다.
3. 노래와 노트가 각각 재생되는 것이므로 싱크가 잘 맞지 않고, 각 컴퓨터의 처리속도마다 노래와 노트의 오차가 생길 가능성이 충분히 존재한다. 이를 해결하기 위해서는 컴퓨터의 계산 시간을 역산해서 단위 기준 시간을 만들어야 오차를 줄일 수 있을 것 같다는 구상을 했다. 하지만 이번 프로젝트를 진행하면서 시간과 능력이 부족했기에 싱크를 맞추는 기능은 구현하지 못했다.
4. Playsound는 .wav파일만 재생할 수 있고 동시에 하나의 소리만 재생할 수 있다. 프로그램 내에 효과음을 넣기 위해 playsound말고 fmod를 사용했다. 하지만 FMOD의 경우 우리 팀에서 작성한 코드로는 대략 1.7이하 버전에서만 실행 가능했기에 호환성에 문제가 있다.
5. 또한 FMOD에 관한 자세한 정보가 적어, 관련 정보를 찾고 프로그램 내에 적용하는 데에 있어 굉장히 오랜 시간이 걸렸고 프로젝트의 시간의 많은 부분을 소진했다. 만약 FMOD에 대한 사용법을 알고 있었다면 프로젝트의 시간 단축이 많이 됐을 것 같다.
6. 인터페이스의 구현을 위해서 프로그램의 흐름을 바꾸는 명령어인 goto문을 많이 사용했다. goto문의 사용을 제외하곤 인터페이스의 반복 및 회구하는 흐름을 구현하는 방법으로 함수를 매번 새로 호출하는 방식밖에 떠오르지 않았고 이 방법은 스택 구조로, 누적되면 굉장히 많은 메모리를 차지할 것 같아서 goto문을 사용했다.
7. 직접 코드를 짜서 프로그래밍을 하는 팀프로젝트의 경우 팀원 모두가 처음이어서, 코드를 공유하고 수정하고 합치는 과정에 있어서 힘들었다. 그리고 함수를 구현하는 과정에서 각자 함수의 리턴 값과 인자, 매개변수 등을 미리 가정하고 구현을 진행했기 때문에, 합치는 과정에서 더욱 힘들었다.
8. **프로젝트 역할 분배**

**시스템 구현 : 전성훈**

* **전체적인 인터페이스**
* 팀원들이 작성한 기능 및 함수들을 인터페이스 내에 이식
* **타이틀화면 구현**
* 시작 화면에서 화살표 방향(key)기반으로 커서 움직이기 구현
* **게임 설명화면 구현**
* **곡 선택 화면**
* 곡 정보 저장 및 곡 정보 출력
* 곡 선택 구현
* equalizer 애니메이션 구현
* 난이도 선택 구현
* **게임 내부 페이지 이동 및 돌아가기 구현**
* 게임 도중 ‘q’누를 시 시작 화면으로의 이동 구현
* 게임 클리어 시, 시작 화면으로의 이동 구현
* 게임 오버 시, 시작 화면으로의 이동 구현
* page변수를 이용하여 함수 별 반환 값 이용해 화면 조정
* **문서화된 파일(txt)을 기반으로 게임 내에서 노트 구현 및 출력**
* 텍스트 파일을 가져온 후 분석해 배열에 저장.
* 노트 2차원 배열 만들어 텍스트파일의 1 값에 따른 노트(“------“) 출력
* 배열의 값을 하나씩 내리면서 노트가 내려오는 효과처럼 보이는 기능 구현
* **게임 요소 출력**
* gameover()화면 구현
* 인게임 내 인터페이스 구현
* 라이프, 코인, 콤보 등을 인터페이스 내에서 출력
* 정확도 판단 값(perfect…)을 인게임 인터페이스 내에서 출력

**게임 요소 구현** : **권혜진**

* **누적콤보 구현**

- 노트 타이밍 perfect, great, good 일 시, 콤보 누적

- 노트 타이밍 ‘bad’일 시, 콤보 0으로 초기화

* **누적 점수 산출**

노트 타이밍 판단에 따른 Perfect (10점) great (6점) good(2점) bad(0점) 점수누적

* **체력(life) 구현 및 게임오버화면**
* 체력은 5칸으로 설정해놓고, 노트 타이밍이 bad일 시, 체력 감소
* 체력 모두 감소 시, gameover()화면 시행
* **게임 클리어 시 랜덤 코인 뽑기**
* 코인이 들어있는 box배열에 0,0,1 임의의 값 지정.
* 난수 생성에 따른 box배열 교환 후 사용자의 입력 값에 따른 랜덤 코인 뽑기 구현
* **게임화면 중 효과 구현**

- 키(a,s,d,f)를 누를 때 마다 “----” 출력 후 제거

- 노트 타이밍 판단, 누적점수, 코인, 콤보 출력 후 제거

* **노트 타이밍 판단 및 출력**
* 노트의 배열에서 타이밍과의 오차 범위에 따른 판단 설정
* 오차범위 +- 1 perfect / 오차범위 +- 2 great / 오차범위 +- 3 good /

노트를 넘어가면 bad처리

* 각 노트 타이밍 판단에 따른 값을 지정하여 변수 judge에 저장
* judge값에 따른 노트타이밍 판단 출력

judge = 1 이면 great / judge = 2 perfect / judge = 3 good / judge = 4 bad

* **코인 사용 유무 판별**
* 코인이 존재할 때 코인 사용유무 묻기

코인 사용 시 – 점수 2배 찬스 사용

코인 사용 하지 않을 시 – 그대로 게임 진행

* **노래마다 노트 타이밍에 맞게 문서화**

DIgga 노래와 Reflection 노래의 easy, normal, hard 난이도에 따른 노트 txt 문서화

* **타이틀화면 중 KW (음표) 구현 (메인화면)**

**노래관련 구현** : **김효민**

* **노래마다 노트 타이밍에 맞게 문서화**
* **FMOD 학습을 통한 노래 재생 및 종료**
* FMOD에 대한 지식이 부족하여 따로 youtube영상을 시청하여 학습을 진행함.

FMOD에 대한 전반적인 내용은 중간보고서에 함께 제출.

https://www.youtube.com/watch?v=xCTIdngHH3Y

* 게임 시작 시, 노래선택에 따른 노래 재생
* 게임 오버 시, 게임 클리어 시, 게임 중단’q’ 누를 시, 노래 중지
* 노래 선택화면 시, 노래선택 (key)값 (화살표)에 따른 노래 변경
* 노래는 melon을 이용하여 다운받음
* **게임 정지 화면, 게임 클리어 화면**
* **랭킹 구현**
* 이진파일을 이용하여 이전의 게임 결과도 누적하여 랭킹을 구현함.
* 이진파일에 저장된 내용을 구조체 배열에 누적하여 저장시킴.
* 순위 정렬 후 이진파일에 저장하는 과정을 거침.