CSED232 Spring 2019 Assignment #4

Music Sequencer

작성자: 정유철 <ycjung@postech.ac.kr>

Due Date: 2019년 5월 11일 토요일 23:59

개요

이 과제물은 C++의 Operator Overloading 을 사용하고, Qt Framework 를 사용해 GUI 어플리케이션 제작을 실습해보는 것을 목적으로 한다.

MUSIC SEQUENCER

Music sequencer 란 음표 (Note)를 조작하여 음악을 작성하고, 편집할 수 있게 해주는 소프트웨어나 하드웨어를 말한다. 이 과제에서는 마우스를 통해 음표를 생성/이동/삭제/재생할 수 있는 피아노 롤 (Piano Roll) 형태의 간단한 Music sequencer 소프트웨어를 직접 구현한다.

Piano Roll

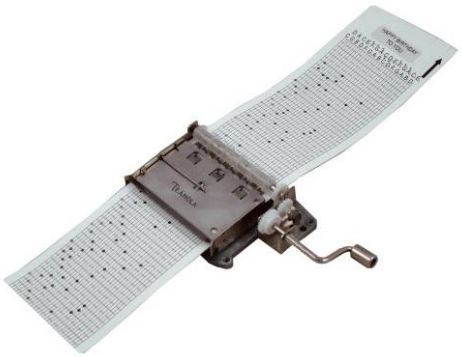


Figure 1 뮤직 박스 위에 2 차원 공간에 점을 찍음으로써 작곡을 할 수 있다. 점 하나는 한 음표의 재생 타이밍과 음 높낮이를 나타낸다. 핸들을 돌리면 종이가 한 쪽으로 서서히 움직이면서 뮤직 박스가 점을 훑는다. 뮤직 박스가 점을 훑는 순간 점의 음 높낮이에 해당하는 음이 재생된다.

피아노 롤은 위의 사진에 나타나 있는 뮤직 박스처럼 2 차원 공간에 음표들을 배치함으로써 작곡을 할 수 있게 해주는 도구이다. 많은 상용 작곡 프로그램은 이와 같은 개념의 피아노 롤 유저 인터페이스를 제공함으로써 사용자가 음표를 배치할 수

있도록 한다. 아래의 그림은 오픈 소스 Digital Audio Workstation 인 LMMS 의 피아노 롤 인터페이스를 보여준다.

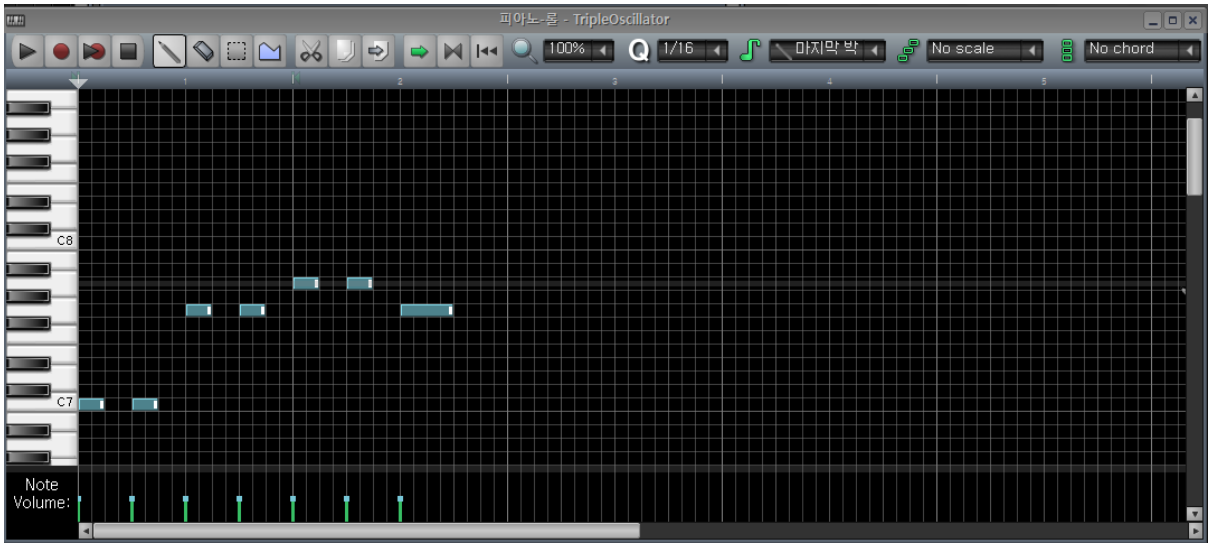


Figure 2 LMMS 의 피아노 롤 인터페이스. 화면에 나타난 푸른 사각형 점들은 음표를 나타낸다. 음표의 세로 위치는 음 높낮이를 나타내고, 음표의 가로 위치는 음의 시작 타이밍을 나타낸다. 음표의 가로 길이는 음이 지속되는 시간을 나타낸다. 실제 작동하는 동영상을 보고 싶다면 (https://youtu.be/7ZHM__Q1bkE) 의 0:40 초 부근을 보면 된다.

프로젝트 설정

MinGW 위에서 Qt IDE 환경 (QtCreator) 설정하기

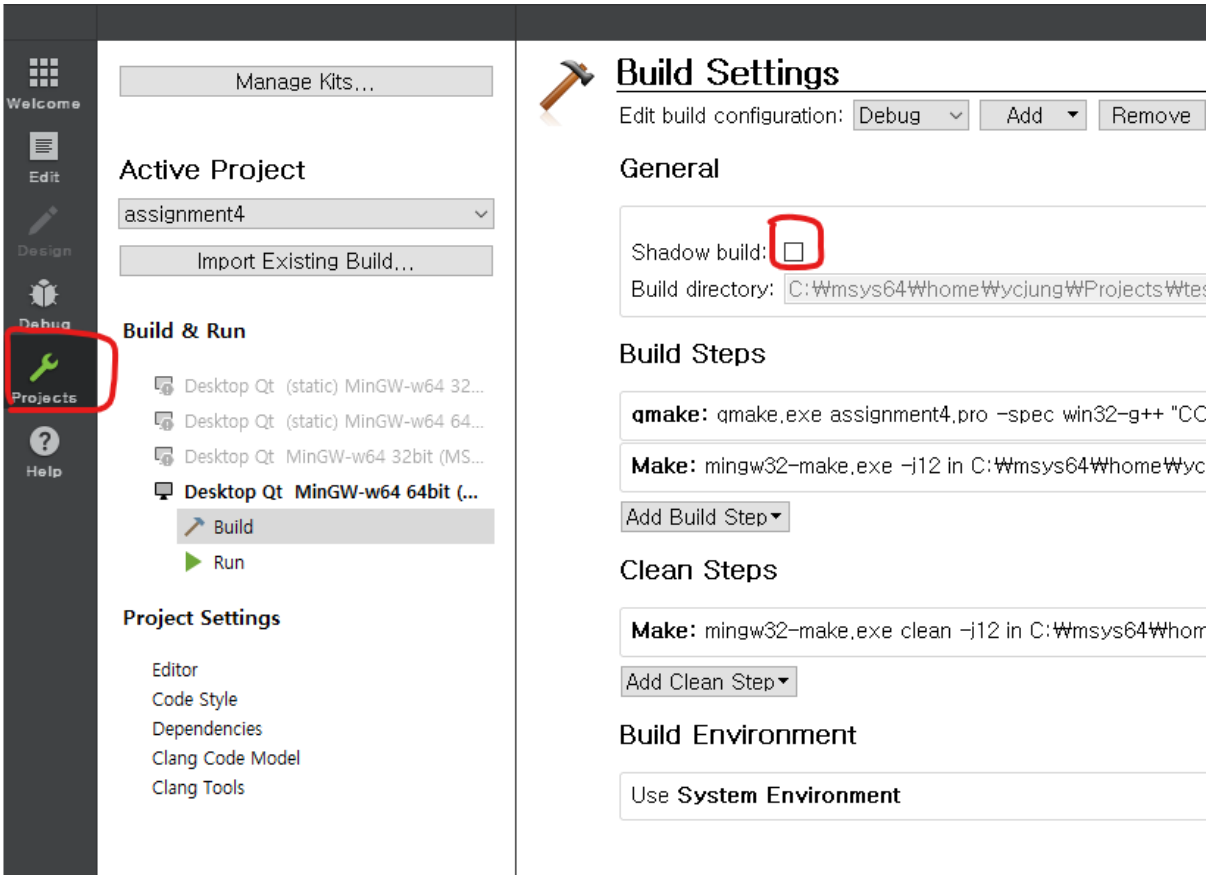
MSYS2 MINGW64 터미널에서, 먼저 모든 설치된 패키지를 최신 버전으로 업데이트한다.

- pacman -Syu

MSYS2 MINGW64 터미널에서 다음 패키지를 설치한다.

- pacman –S mingw-w64-x86_64-toolchain
  - C++ 컴파일러와 디버깅 툴
- pacman –S mingw-w64-x86_64-qt5
  - Qt5 프레임워크
- pacman –S mingw-w64-x86_64-qt-creator
  - QtCreator IDE

설치 완료 후 터미널에서 "qtcreator &" 명령어를 실행하여 IDE 를 실행한다. 제공된 스켈레톤 코드를 다운받고, QtCreator 의 File -> Open File or Project 를 통해 assignment4.pro 를 연다. Toolchain 을 선택하라는 대화창이 나오면 Configure 버튼을 눌러 다음으로 넘어간다. 그 후 Projects 탭에서 "Shadow build" 체크박스를 해제한다.



설정이 끝나면 왼쪽 아래에 있는 망치 모양 아이콘을 클릭해 빌드가 잘 되는지 확인한다.

프로젝트 파일 구성 설명

본 과제물은 수강생이 음악 이론에 대한 사전 지식이 부족할 수 있음을 고려하여, 음악과 관련된 데이터 표현에 대한 기본적인 스켈레톤 코드를 제공한다. 코드 사용 예시는 test_model.cpp 를 참조 바람. 스켈레톤 프로젝트는 "app", "as4", "test_io", "test_model" 4 개의 하위 프로젝트로 구성되어 있다.

- "app"
  - 우리가 최종적으로 만들고자 하는 GUI 프로그램 실행파일을 빌드한다.
    - main.cpp – main() 함수가 있는 소스 코드

- "as4"
  - "app"과 테스트 코드에서 공통으로 쓰이는 정적 라이브러리를 빌드한다.
    - model/time.h(.cpp) – 작곡에 사용되는 시간 단위와 단위 간 변환에 대한 코드를 제공한다.
    - model/pitch.h(.cpp) – 음 높낮이를 나타내는 클래스와, 해당 클래스를 문자열로 변환할 수 있는 편의 함수를 제공한다.
    - model/note.h(.cpp) – 음표 하나를 나타내는 클래스를 제공한다.
    - model/seq.h(.cpp) – 트랙 하나에 해당하는 음표들의 시퀀스에 대한 클래스를 제공한다
    - model/song.h(.cpp) – 여러 트랙으로 이루어진 곡 하나를 표현하는 클래스를 제공한다.
    - io/seqio.h(.cpp) – 트랙 하나에 해당하는 음표들의 시퀀스에 대한 입출력 함수를 선언한다. (구현 없음)
    - io/songio.h(.cpp) – 곡 하나에 대한 입출력 함수를 선언한다. (구현 없음)
    - util/require.h – 프로그램 작성시 사용할 수 있는 편의 함수 제공
    - widgets/mainwindow.h(.cpp) – QWidget 을 사용한 메인 GUI 창에 대한 선언.
      - 다른 widget 생성시에도 widgets/ 하위 폴더 안에 넣는 것을 추천
- "test_io" (초기 상태에는 프로젝트에 포함되어있지 않음)
  - 파일 입출력과 관련된 유닛 테스트를 빌드한다.
- "test_model"
  - 본 프로젝트에 기본으로 주어지는 라이브러리 코드에 대한 유닛 테스트를 빌드한다.

- assignment4.pro
  - 전체 프로젝트 설정 파일. 이 프로젝트 파일 자체는 거의 하는 것이 없고, qmake/ 디렉터리 아래 있는 하위 프로젝트를 불러와주는 역할만을 한다.

- qmake/as4/as4.pro
- qmake/app/app/pro
- qmake/test_io/test_io.pro
- qmake/test_model/test_model.pro

코드 작성 관련 주의사항

- 새로운 .h / .cpp 파일을 만들면 **반드시** 해당하는 하위 프로젝트의 .pro 파일에 소스 코드와 헤더를 명시해줘야 한다. 이 과정을 지나칠 시 해당 소스 코드가 빌드 되지 않는다.
- Qt 의 QObject 를 상속하고, Q_OBJECT 매크로를 사용한 클래스의 선언과 구현을 한 .cpp 파일 안에 했다면, 반드시 <.cpp 파일 이름>.moc 을 파일 맨 아래에 include 해야 한다.
- 만약 소스 코드를 제대로 작성한 것 같은데도 컴파일/런타임 오류가 발생하면 프로젝트를 Clean 후 Rebuild All 하는 것이 도움이 될 수 있다.

요구사항

각 문제 옆 비중은 AssnReadMe.pdf 의 "프로그램 기능", "프로그램 설계 및 구현"을 합친 점수에 대한 비중을 의미함.

Problem 1: Operator Overloading (30%)

assignment4.pro 의 맨 아래 두 줄을 주석해제하면 (맨 앞의 "#" 문자를 삭제), test_io 프로젝트가 다음 빌드부터 컴파일된다. 그러나 소스 코드는 컴파일이 되지 않는다. 테스트 코드 (tests/test_io.cpp) 를 수정하지 말고, io/ model/ 의 소스코드를 수정하여 모든 테스트를 통과시키자.

테스트 파일은 빌드 후 build/test/test_io.exe 에 있다. MINGW64 터미널로 실행하거나, QtCreator 의 좌측 하단 모니터 아이콘을 클릭하여 실행할 타겟을 test_io 로 전환한 뒤, 초록 화살표 아이콘을 눌러 실행할 수 있다.

소스코드에서 사용하는 QCOMPARE 매크로에 대한 설명은 Qt Test 공식 문서에서 읽을 수 있다. (https://doc.qt.io/qt-5/qttest-index.html) 내부적으로 "==" 연산자를 사용한다.

채점 기준

- 빠져 있는 operator 에 대한 옳은 선언 : 10%
- 빠져 있는 operator 에 대한 옳은 구현 : 20%

Problem 2: Player 및 하위 클래스 디자인/구현 (30%)

스켈레톤 코드는 음표 시퀀스를 조작할 수 있을 뿐 재생의 개념이 구현되어 있지 않다. **자신이 GUI 에서 어떤 방식으로 해당 클래스를 사용할지 고려하며** as4::model::ISeq 를 재생할 수 있는 클래스를 한 Song 을 이루는 트랙 별로 (총 2 클래스. Melody 트랙과, Drum 트랙) 적절히 디자인하고, 구현하자. 클래스의 이름을 포함한 정의는 자유롭게 할 수 있다.

각 음표에 대해 재생되는 오디오는 스켈레톤과 함께 주어지는 .wav 파일들을 사용한다. (audio/melody, audio/drum). 각 파일의 파일명은 다음과 같이 구성되어 있다. pitch_class 와 octave 에 대한 정의는 model/pitch.h 에서 읽을 수 있다.

<pitch_class>_<octave>.wav

두 트랙은 다른 재생 방식을 가진다.

1. Melody 트랙

   Note 의 GetStart() 값에 해당하는 시간에 오디오 재생을 시작하여, GetDuration() 값에 해당하는 유지 시간이 끝나면 음표의 오디오 재생을 곧바로 멈춘다.

2. Drum 트랙

   Note 의 GetStart() 값에 해당하는 시간에 오디오 재생을 시작하여, GetDuration() 값에 해당하는 유지시간을 무시하고 음표의 오디오 음원을 끝까지 재생한다.

만일 재생하고자 하는 음에 해당하는 오디오 파일이 없다면 아무 소리도 내지 않는 것으로 한다.

오디오 파일과 같은 리소스들을 Qt 프레임워크에서 관리하는 방식에 대해서 다음의 공식 문서를 참조하면 배울 수 있다. (https://doc.qt.io/qt-5/resources.html)

오디오 파일을 실제로 재생하는 것은 Qt 프레임워크의 Audio Overview 문서를 참조하면 배울 수 있다. (https://doc.qt.io/qt-5/audiooverview.html) QSoundEffect 클래스는 .wav 파일을 재생하는 직관적인 인터페이스를 제공한다. (QSoundEffect::play(), QSoundEffect::stop() 슬롯 참조.)

Audio 등 Qt 추가 기능을 사용할 때, as4.pro 파일에 "QT += multimedia" 처럼 QT 변수에 추가 기능의 모듈 이름을 추가해줘야 함에 유의한다. 사용할 클래스에 해당하는 Include 문과 QT 변수에 들어갈 모듈 이름은 Qt 클래스 레퍼런스 문서에 명시되어 있으니 참조 바람. (예시: https://doc.qt.io/qt-5/qsoundeffect.html)

일정 시간 간격으로 특정 Slot을 호출하는 것은 QTimer를 사용하면 간단하다. 해당 문서 참조 바람. (https://doc.qt.io/qt-5/qtimer.html#QTimer). 해당 클래스를 사용한 아날로그 시계 구현 예시도 제공되니 참고 바람. (https://doc.qt.io/qt-5/qtwidgets-widgets-analogclock-example.html)

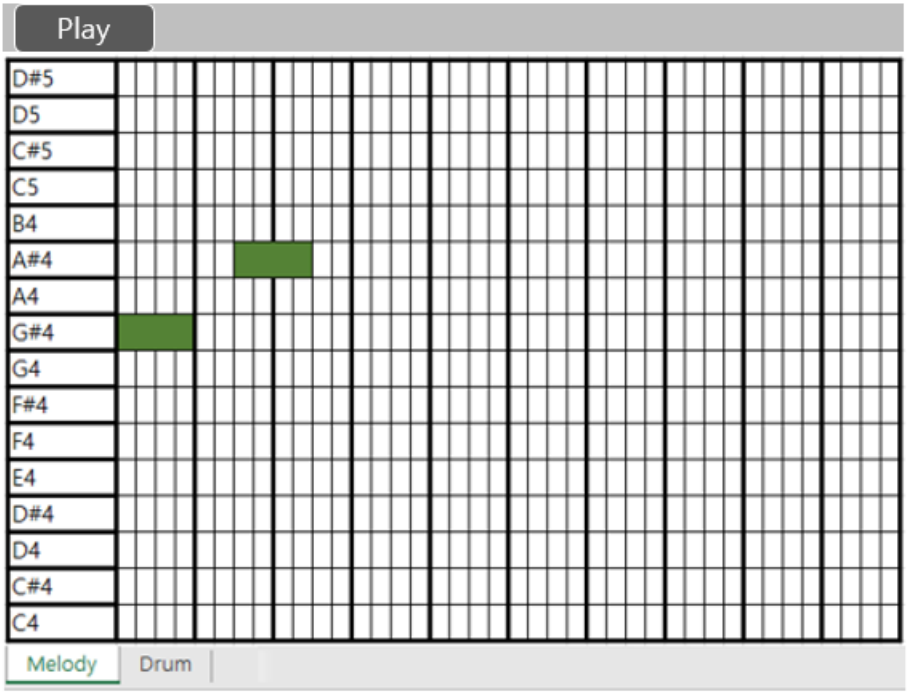
음표에 사용되는 시간 단위와 실제 초 단위 간 변환은 model/time.h 에 이미 구현되어 있으므로 사용을 권장함.

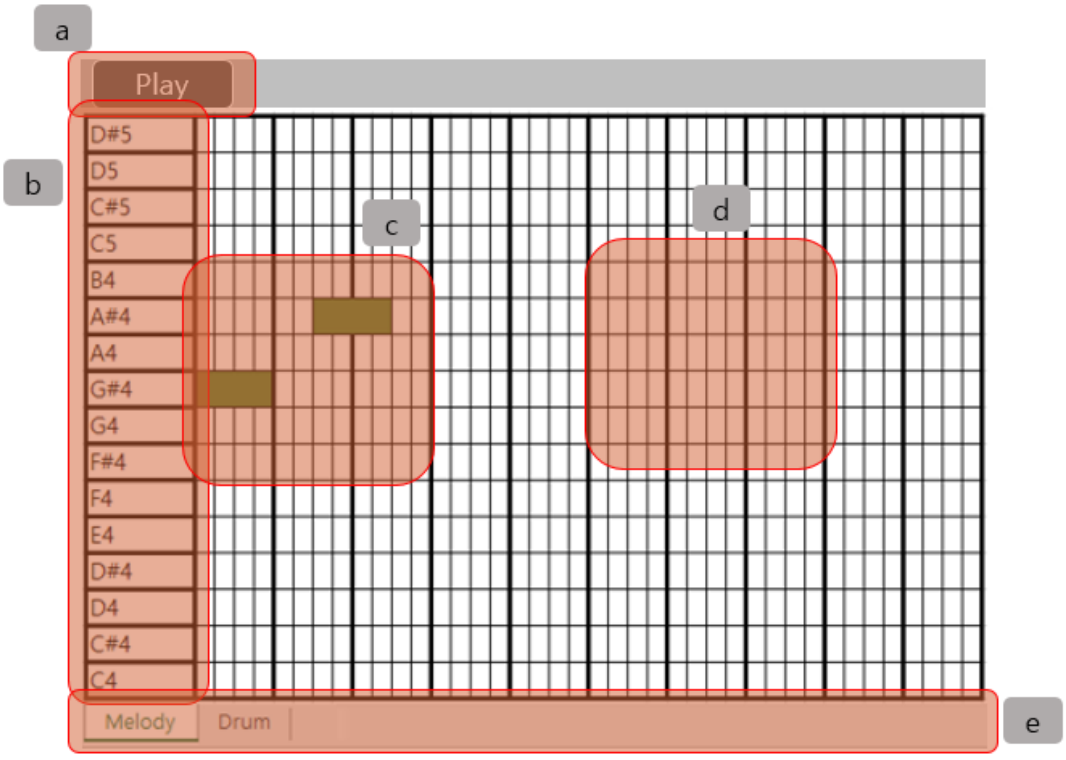
채점 기준

- 트랙 별 재생 방식 차이를 클래스 상속 관계와 Polymorphism 을 이용하여 코드 복사-붙여넣기 없이 올바르게 구현 : 30%
    - 단순 bool 플래그 사용해서 구현 시 감점 : 현재는 과제의 단순화를 위해 사소한 차이점만 주었지만, 실제 상황에서의 다른 트랙 구현이나 기능 확장을 생각해 보았을 때 클래스 상속을 활용하는 것이 정당함.
- 오디오 재생 속도 및 싱크는 꼭 정확하게 맞지 않아도 상관 없다. 뒤에 오는 음표보다 앞에 오는 음표가 먼저 재생되는 정도로 충분하다.

Problem 3: GUI Programming (40%)

Qt Framework 를 활용하여 다음과 같은 GUI 를 구현한다. 첫 번째 그림은 최종 GUI 예상도이며, 두 번째 그림은 각 UI 의 구분을 나타낸 그림이다.





UI 요구사항

a. Play – 현재까지 작업한 내용을 오디오로 플레이한다. 재생 속도는 model/time.h 에 정의된 as4::model::time::DefaultConfig() 에 해당하는 120BPM 을 기준으로 한다. 플레이 중 노트 수정사항은 실시간으로 오디오에 반영하지 않아도 좋다. 이미 오디오가 플레이 중인 경우 Play 버튼을 클릭해도 반응하지 않는다. 오디오 플레이 중에 UI 가 무반응 상태가 되어도 상관 없다. 또한, 마지막 beat (10 번째 beat) 가 끝날 때 아직 재생될 오디오가 남아있다면 (예 : Drum 트랙의 일부 Note 또는 오른쪽 경계 밖으로 벗어난 Note), 바로 재생을 멈춰도 상관 없다.

b. 각 가로줄에 해당하는 음계 (pitch_class)를 텍스트로 표시한다. 스크롤 바 없이 C4 부터 D#5 까지만 표현하면 된다.

   A. model/pitch.h 에 관련 편의 함수가 구현되어 있으니 참고 바람

c. Note. Note 하나에 해당하는 직사각형 안쪽을 마우스 오른쪽 클릭하면 해당하는 Note 를 삭제할 수 있다. UI 에서 추가하고 편집하는 Note 는 항상 1 beat 의 길이를 가지며, 이 길이를 수정하는 기능은 구현하지 않아도 된다. 한 번 생성된 Note 의 위치를 바꾸는 기능 또한 구현하지 않아도 된다.

d. Piano roll. 각 얇은 세로줄은 model/time.h 에 설명된 1 unit 을 의미하며, 굵은 세로줄은 1 beat 를 의미한다. 1 beat 는 4 unit 으로 이루어져 있다. 빈 공간에 마우스 왼쪽 클릭하면 클릭 위치와 가장 가까운 unit 에서 시작하는 길이 1 beat 의 새 Note 가 클릭 위치에 해당하는 음계에 생성된다. Note 는 Note 가 가지는 음계에 해당하는 세로줄에 위치한 가로 길이 4 칸의 직사각형으로 나타난다. 새 Note 생성시 음계에 해당하는 오디오를 재생할 필요는 없다. 세로줄은 스크롤 바 없이 10 beat 까지만 화면에 표시된다. 즉, 굵은 세로줄로 나누어진 칸 10 개만 UI 에 표시된다.

e. 트랙 전환 탭. 현재 편집 대상으로 활성화된 시퀀스를 Melody 와 Drum 사이에서 전환한다. Melody 버튼을 클릭하면 Melody 시퀀스가 Piano roll 에 나타나고, Drum 버튼을 클릭하면 Drum 시퀀스가 Piano roll 에 나타난다. 트랙 전환은 편집 내용을 삭제하지 않는다. (예: Melody 시퀀스에 Note 2 개를 추가했다가 Drum 으로 전환 후 다시 Melody 로 돌아오면 이전에 추가했던 Note 2 개가 같은 위치에 보여야 한다.) 프로그램 실행 시에는 Melody 가 기본 편집 대상으로 활성화된다.

채점 기준

- a – 10%
- b, e – 10%
- c, d – 20%
- 위의 UI 요구사항에 명시되지 않은 기능은 구현하지 않아도 된다.
  - 예) Note 길이 수정 기능, BPM 수정 기능 등
- UI 의 스타일은 점수에 반영하지 않는다.
  - 예) 버튼 색깔, 디자인, 글꼴, 글자 크기 등

스켈레톤에 사용된 STL 클래스 관련 사항

현재 스켈레톤 코드에는 STL 클래스인 std::vector 와 std::tuple 가 사용되고 있음. std::vector<T>는 T 타입을 원소로 지니는 1 차원 가변배열을 나타냄. std::tuple<A,B,C>는 A, B, C 타입의 값을 순서대로 묶은 값을 나타냄. 아직 이 부분에 대한 자세한 이해는 필요하지 않으나, 테스트 코드를 이해하는데 설명이 필요하다면 이에 대한 인터넷 문서 참고 바람.

(https://en.cppreference.com/w/cpp/container/vector)
(https://en.cppreference.com/w/cpp/utility/tuple)

스켈레톤 수정 관련 사항

스켈레톤은 자유롭게 수정 가능하나, test_io.cpp 는 수정할 수 없음. 수정 후 해당 테스트가 잘 돌아가는 것을 확인 바람.

제출 유의사항

- 이번 과제는 빌드 시스템이 스켈레톤으로 주어지므로 따로 Makefile 은 만들 필요가 없음. 컴파일이 되는 것을 확인한 프로젝트 폴더를 Clean 후 그대로 압축해서 제출할 것.
  - 채점 환경은 MinGw 상 QtCreator 사용할 것임
- 학기 초에 공지된 AssnReadMe.pdf 를 반드시 읽고 제출할 것.