2019년-Spring

객체지향프로그래밍

Assignment #4

학번: 20180085

학과: 무은재학부

이름:송수민

POVIS ID: songsm921

명예서약(Honor code)

"나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다."

Assignment #4 : Music Sequencer 구현

#1. 알맞은 Operating Overloading 구현

I. 프로그램 개요

본 과제 1번 문항은 필요한 Operator가 구현되지 않아, 원래의 Operator와 다른 Operator를 Overloading하는데 목적이 있다.

II. 프로그램 구조 및 알고리즘

Test.io 프로젝트를 열면, 소스파일을 Run해보면 알맞은 Operator가 존재하지 않아, Build가 불가능하다. 본 과제에서 필요한 Operator Overloading은 크게 3가지이며, 세부적으로 나누면 총 6개이다. 그 Operator들은 아래와 같다.

1. <<(Shift Operator)

- ostream& operator<<(ostream& , const ISeq&); // Seqio

- ostream& operator<<(ostream& ,const Song&); //Songio

2. >>(Shift Operator)

- istream& operator>>(istream& , ISeq&); // Seqio

- istream& operator>>(istream& ,Song&); //Songio

3. ==(Comparison Operator)

- friend bool operator==(const ISeq&, const ISeq&);

- friend bool operator==(const VectorSeq&, const VectorSeq&); // Seq

1. << (Shift Operator)

이 Operator는 ISeq, Song과 ostream간의 파일 입출력을 매개해주는 operator이다. 이 operator 내부에는 Assignment 2,3의 파일 입출력 부분을 참고하여 작성하였다. ISeq의 경우 Melody나 Drum과 같은 구별할 것이 필요하지 않아, 각각의 정보 사이를 공백으로 구분하여 작성하게 구현하였다. Song은 Melody와 Drum을 구분할 필요가 있어서 Melody의 정보를 모두 입력하면 Fin을 작성하고 나서 Drum에 대한 정보를 입력하게 작성하였다.

2. >> (Shift Operator)

이 Operator는 ISeq, Song과 istream간의 파일 입출력을 매개해주는 operator이다. 이 부분도 1과 마찬가지로 Assignment 2,3의 파일 입출력을 참고하여 작성하였다. ISeq의 경우 구분할 정보가

존재하지 않아, 차례로 정보를 입력하게 작성하였고, Song의 경우 Melody와 Drum을 구분 할 필요가 있어 처음에는 Melody의 정보로 계속 입력하다가, Fin이 적혀있는 Line을 받으면 mode를 2로 변경하고 continue한다. Mode를 2로 변경하였기 때문에, 다음 line부터의 정보는 Drum의 정보로 입력된다.

3. ==(Comparison Operator)

QCOMPARE 내부적으로 사용하는 == Operator에서 ISeq 간, VectorSeq간을 비교해주는 Operator가 존재하지 않아 이를 Overloading하였다. 비교 연산자이기 때문에, 각 클래스 객체안의 정보를 비교해 주어야 하는데, 코드 작성 편리함을 위해 5개의 Method를 추가하였다.

```
unsigned Getsize()const override;
TimeStamp Enter1(unsigned i)const override;
TimeInterval Enter2(unsigned i)const override;
int Enter3(unsigned i)const override;
 int Enter4(unsigned i)const override;
```

각각 Vectorseq안의 m_notes의 정보를 받기 위한 Method이며, 차례로 m_notes의 size를 return, m_notes가 가리키고 있는 start, duration, pitch_class, octave를 각각 return해준다.

이 Method를 통해 Return된 값을 서로 비교하여 QCOMPARE을 진행한다. Top-level Function을 사용하기 위해 Seq.h에 선언하고 Friend함수로 선언하였다.

III. 프로그램 실행 방법 및 실행 결과

```
Config: Using QtTest library 5.12.2, Qt 5.12.2 (x86_64-little
PASS   : TestIo::initTestCase()
PASS   : TestIo::ReadWriteSeq()
PASS   : TestIo::ReadWriteSong()
PASS   : TestIo::cleanupTestCase()
Totals: 4 passed, 0 failed, 0 skipped, 0 blacklisted, 16ms
********* Finished testing of TestIo *********
```

위 사진은 모든 Operator를 구현하고, testio를 Run하였을 때 결과이다. 모든 Test를 Pass한 것을 확인 할 수 있다.

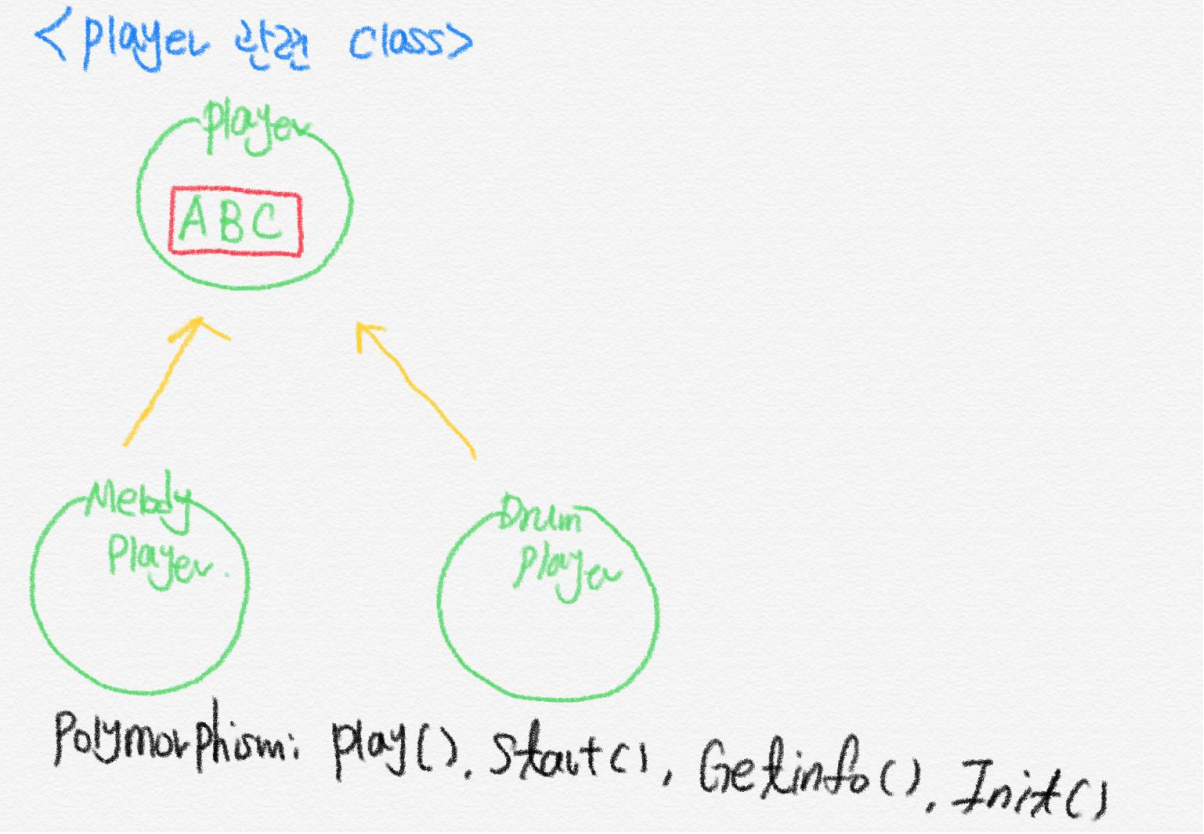
#2 & 3. Music Sequencer 구현

I. 프로그램 개요

본 과제 2번과 3번 문항은 주 목적인 Qt를 사용하여 Music Sequencer의 주 기능인 재생 및 정지를 구현하고 이에 대한 GUI를 구현하는 것이다.

II. 프로그램 구조 및 알고리즘

Player를 구현하기 위해 Player라는 추상 클래스를 선언하고 그에 따른 MelodyPlayer와 DrumPlayer라는 클래스를 Player를 상속하여 선언하였다. 클래스의 구조는 아래 그림과 같다.



MelodyPlayer와 DrumPlayer의 알고리즘은 아래와 같다.

1. MainWindow에 있는 TabWidget 1,2에 각각 Play 버튼을 누르면 탭에 해당하는 Player가 호출된다. 아래부터는 MelodyPlayer로 설명하기로 한다.

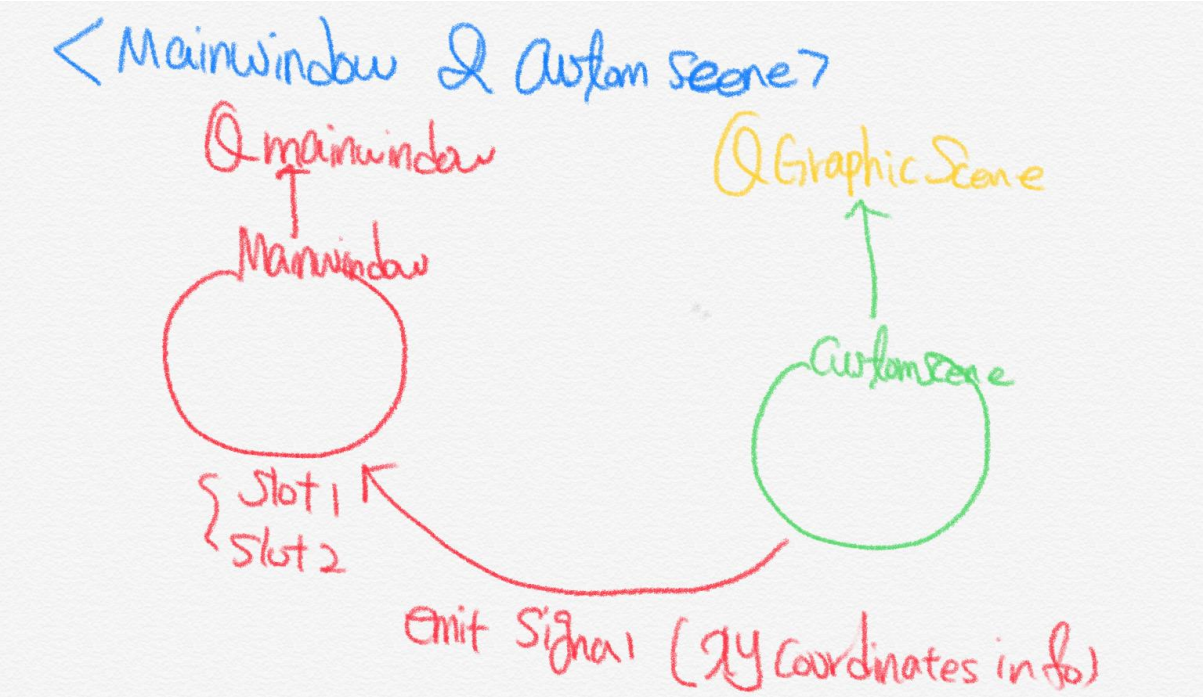
2. Play버튼을 누르면 MelodyPlayer에 있는 Getinfo 함수에 ISeq의 포인터 객체가 전달된다. 그 객체에서 필요한 정보를 Player내부에 있는 Vector에 저장한다. 저장하는 정보는 아래와 같다.

- Note의 Start Unit Value, Start Unit Value + Duration Unit Value를 더한 Stop Value

- Note의 pitch_class, octave value

3. Getinfo()가 실행 된 후, Init()을 실행하면 timer가 실행되며 재생 및 정지 과정이 시작된다. 1 unit이 0.125초이기 때문에, timer는 0.125초 마다 timeout시그널을 발생시키고 이에 전체 시간 축을 담당하고 있는 LineX의 접근 인덱스를 1 증가시키며 그 인덱스에서 시작해야 하거나, 끝내야 하는 Note가 존재한다면 재생 시에는 그 Note의 pitch, octave 값을 Play()로 넘겨 이 조건에 맞게 재생하고, 정지한다면 그 플레이어의 stop을 호출한다. (Melody에만 정지과정이 존재한다.) For문의 연산속도가 매우 빨라, 화음처리도 가능하다. 이때, index는 무조건 1부터 시작하기 때문에, index가 0인 부분을 처리를 하지 못하는데, 이는 Init에서 0인 부분에 대한 예외처리를 하였다.

다음은 Music Sequencer의 GUI 구현 부분이다.



MainWindow안에 두개의 TabWidget이 있고, 각각 Melody Tab과 Drum Tab을 담당한다. 각각 다른 Tab이기 때문에 탭을 변경하여도 원래의 탭은 변경사항이 없다. 각각의 Tab이기 때문에, 각각 GraphicView를 배경으로 사용하였다. 마우스 이벤트를 받는 것은 GraphicsScene을 사용하였다. 생성되는 사각형 그래픽 아이템은 GraphicRectItem을 사용하였다. 노트 입력 과정은 아래와 같다.

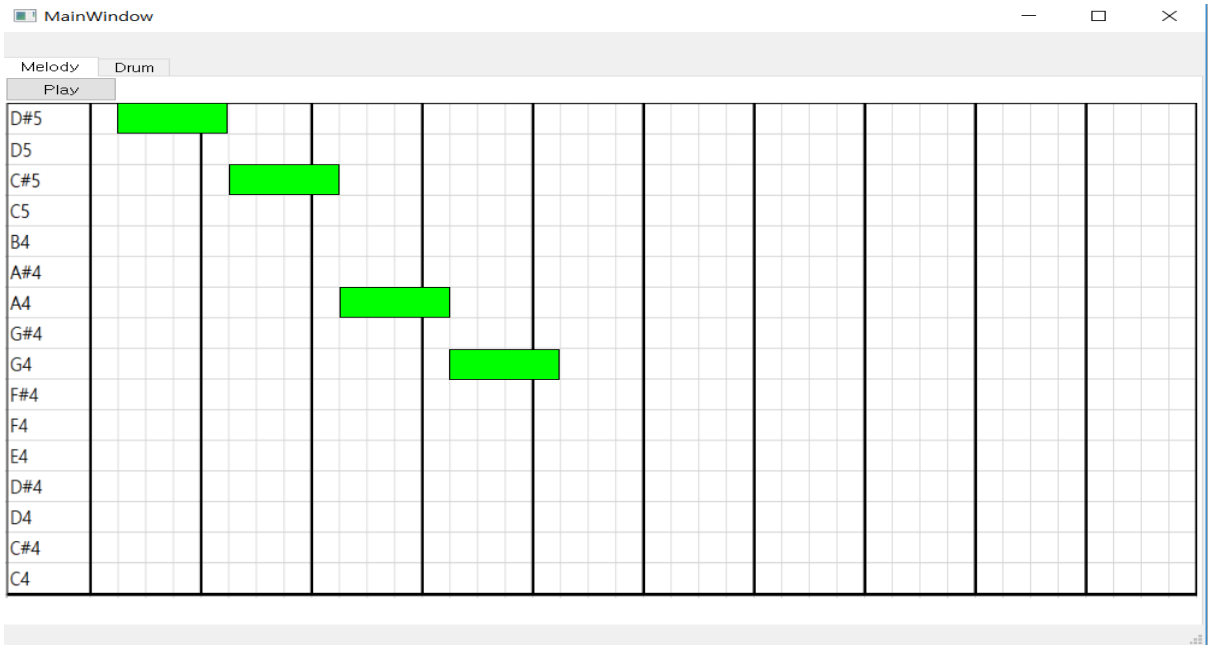
1. CustomScene에서 마우스 좌표 및 좌클릭인지 우클린인지를 판별하여 이에 해당하는 정보를 MainWindow 클래스로 넘긴다.

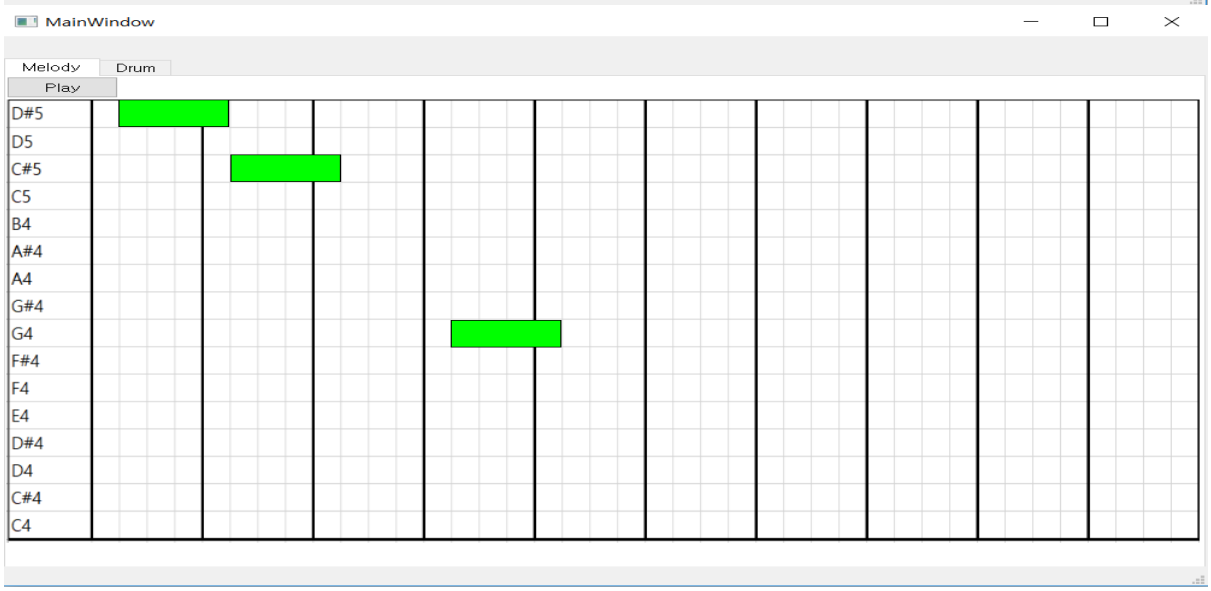
2. Connect로 연결하였기 때문에 이 정보는 각 Slot으로 전달되고, 다음 과정을 처리하기 전에 먼저 사용자가 클릭한 좌표를 if – else if문으로 보정한다. 보정한 값으로 해당하는 Note 시작 부분, 사각형 생성 좌표, pitch, octave 모두를 결정한다.

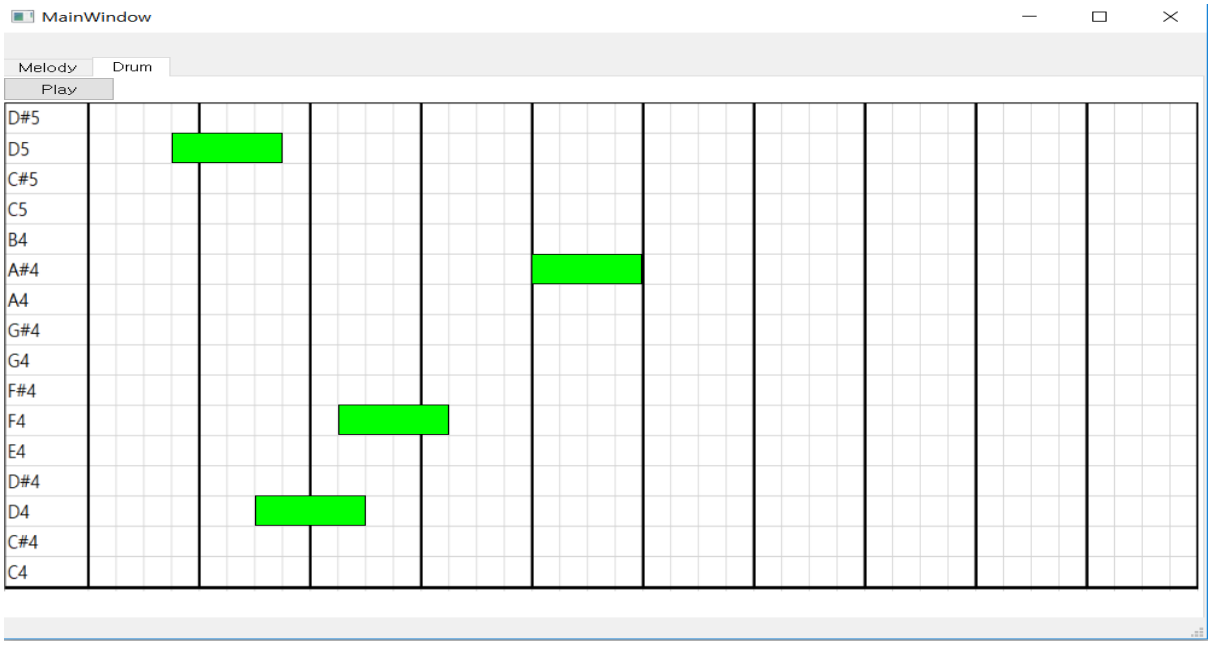
3. 좌클릭이면 미리 보정한 좌표 정보로 사각형 아이템을 생성하고, 그 좌표의 값을 후에 삭제 과정을 위해 Vector에 저장한다. 그 후, ISeq에 저장한다.

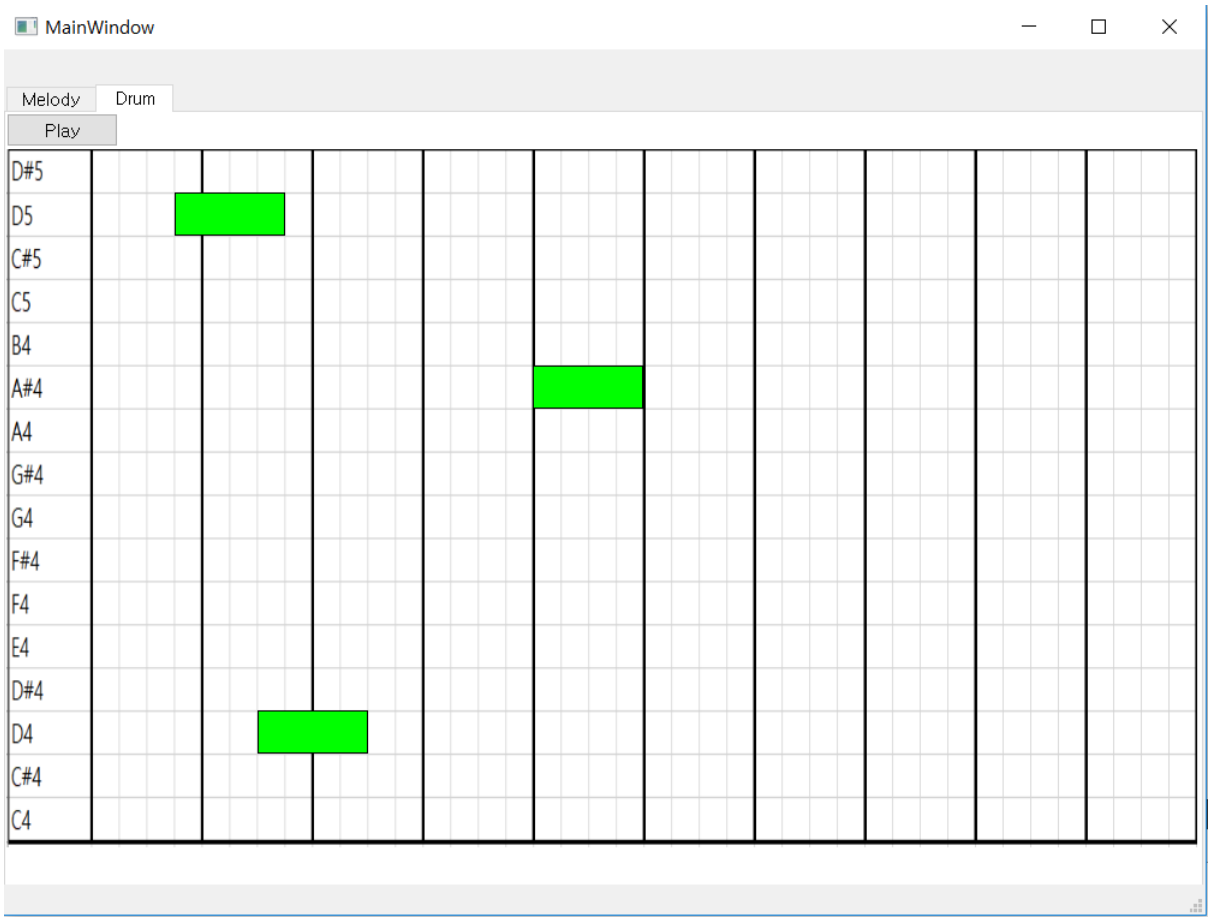
4. 우클릭이면 미리 보정한 좌표 정보와 저장해둔 좌표 정보를 비교하여, 일치하는 정보가 존재한다면 그 좌표정보로 해당하는 사각형의 위치를 보정하고 삭제한다. ISeq에서도 삭제가 필요하기 때문에 정보를 넘겨 삭제 과정을 진행한다.

III. 프로그램 실행 방법 및 실행 결과









차례로 Melody, Drum Tab에서 생성 후 삭제 과정을 1회 실행한 사진이다.

IV. 토론

굉장히 어려웠다. 새로운 개발환경에서 과제를 수행하다 보니 예기치 못한 오류들이 많이 발생하였고, 특히 조작하지 않은 곳에서 오류가 발생하였다고 하였을 때 굉장히 곤혹스러웠다. 또한 QT는 버그가 존재하는 듯하다. 친구들이 많이 사용하는 QSoundEffect가 본인 컴퓨터에서는 작동하지 않아 꽤 오랜시간을 허비하였다. 또한, 처음 다루어 보는 GUI는 상당히 난이도가 있었다. 특히, Connect부분과 View-Scene-Item 간의 좌표축 설정이 특히 어려웠다. View의 좌표의 시작 부분은 좌측 상단 부분이었는데, Scene의 좌표의 시작은 화면 중앙이었다. 이에 따라, 좌표를 동일하게 맞춰주는 작업이 필요하였다. 또한, Item의 좌표는 예상했던 것과 달리, 자기 자신이 (0.0)이 되어 삭제 과정 중 이를 보정하는 작업이 필요하였다.

V. 결론

상당히 어려웠지만 Qt Creator가 GUI를 제작하는데 쉽다는 것에 동의 할 수 있게 되었다. 여러 시행착오를 거치며 GUI를 제작하는 능력을 배양 할 수 있었던 과제였다. 또한, 여러 프로젝트 파일이 존재하는 과제를 수행하였는데, 이를 유기적으로 활용하는 법도 알게 되었다.

VI. 개선사항

1. For문을 이용한 화음 처리

이는 for문의 연산속도를 이용한 처리라고 할 수 있는데, 염연히 말하자면, 동시에 재생되는 것은 아니다. 과제를 수행한 후, 이를 처리하는 방법을 생각해 보았는데, for문을 사용하되, 어떤 시점에 재생 되어야하는 수를 카운트 한 후, 그 만큼의 객체를 생성하여 동시에 재생하는 방법을 생각해보았다. 이 방법을 적용한다면 현재 Play를 처음 눌렀을 때, 재생이 원활히 되지 않는 문제가 발생하기도 하는데, 이 문제를 해결할 수 있을 것이라 생각한다.

2. 지나치게 긴 if-else if문을 사용한 좌표 보정 과정

본래는 각 unit간의 간격이 동일하기 때문에, 수열 식과 /연산자의 몫이 처리 되는 것을 적절히 섞어 일반항을 만들었으나, 생각대로 되지 않았고, 이를 분석해본 결과 해당하는 좌표가 정확히 맞지 않는다는 것을 알아 이를 하나하나 조정해주게 되었다. 그리 한 결과, 코드의 한 부분이 길어지게 되었다. 정확성을 얻는다는 이점은 있으나, GraphicView를 설정할 때, 이를 고려한다면 정확성과 코드의 가독성 모두 얻을 수 있을 것이라고 생각한다.