Window 환경에서의 악보 생성기의 구현

o
전종욱, 박성순
안양대학교 컴퓨터학과

Implementation of Musical Score Generator on the Window Environment

o
Jong-Uk Jeon, Sung-Soon Park
Dept. of Computer Science and Engineering, Anyang University

요 약

본 논문에서는 악보를 함께 출력함으로써 악기를 배우거나 노래를 연습하도록 도와주는 악기학습 및 노래연습 프로그램에서 사용되는 악보를 현재 미디 정보의 저장에 보편적으로 사용되고 있는 SMF 형식으로 저장된 미디 파일을 이용하여 생성하는 방법과 그 구현 방법을 제시한다.

1. 서론

최근 음악작업에 관련된 컴퓨터 하드웨어 및 소프트웨어 발달과 미디(MIDI) 장비의 보급으로 인해 이 같은 환경을 갖추어 집에서 자신이 직접 작곡 및 편곡을 하는 사람들이 점차 늘고 있다. 또한 이러한 사람들에 의해 만들어진 음악 파일들은 인터넷을 통한 자료의 공유가 확대됨에 따라 여러 사람들에게 널리 퍼져 그 종류가 다양하고 수량도 많아 졌다.

사람에게 있어서 음악이란 생활의 일부분이라고도 할 수 있을 만큼 아주 밀접한 관계를 가지고 있다. 이러한 관계로 사람들은 누구나 노래를 잘 부르거나 악기를 잘 다루고 싶은 욕구를 가지며 이를 위해 부단히 노력한다.

노래를 배우거나 악기를 연습하는 방법의 기초 방법 중 하나는 악보를 보며 연습하는 것이다. 아무리 같은 곡을 반복해서 듣는다 하더라도 악보를 보며 따라 하는 방식과는 비교할 수 없다. 그렇기 때문에 현재 많은 사람들은 자신이 배우려는 곡의 악보를 얻기 위해 그 악보가 실린 노래집이나 악보집을 사고 있다. 그러나 자신이 필요로 하는 곡을 찾기엔 그리 쉽지 않으며 얻기 위한 대가도 적지 않게 지불해야 하는 실정이다.

본 논문은 이같이 불편한 상황을 개선하는 방안으로 현재 많은 사람들에게 보편적으로 사용되고 있는 음악 파일인 SMF(Standard MIDI File) 형식의 파일을 이용하여 악보를 출력하는 프로그램의 구현 방법과 그 개발 과정을 기술한다.

2. MIDI 파일의 구조

2.1 SMF 형식의 기본 구조

서로 다른 연주 장비들 사이에서 연주 데이터를 상호 이용하기 위해서는 공통으로 사용할 수 있는 데이터 형식(Data Format)이 필요하다. SMF 는 이러한 미디 규격의 하나로 현재 주로 사용되는 형식이며 일반 PC 에서는 *.mid 라는 확장자명으로 저장된다. 또한 SMF 는 저장되는 트랙(Track)의 성격에 따라 Single Multi-channel Track 과 Simultaneous Multi-Track, 그리고 Sequentially Independent Single-Track 의 3 가지 형식을 가진다. 이하 본 논문에서 미디 파일이라 함은 SMF 형식으로 저장된 미디 파일이라 하고 두 번째 형식인 Simultaneous Multi-Track 을 사용한다.

미디 파일은 여러 개의 청크(Chunk)들로 이루어지는데 청크는 헤더 청크(Header Chunk)와 트랙 청크로 구분되며 각 청크들은 청크 식별자와 청크의 길이, 실제 트랙 데이터들로 구성된다. 다음은 미디 파일 내의 청크들이 포함된 형태와 청크의 기본 구조를 나타낸 것이다.

```
[Syntax of a MIDI File] ;
<Header Chunk>
<Track Chunk>
<Track Chunk>
        :

[Syntax of a Chunk] ;
<chunk type> <length> <track data>
```

헤더 청크는 파일의 맨 처음 부분에 단 한 개만 존재하는 것으로 미디 파일 형식과 헤더를 포함한 트랙의 수 등과 같은 일반적인 전체 파일의 정보를 담고 있다. 트랙 청크는 트랙의 수 만큼 존재하며 각 트랙의 길이와 실제 미디 데이터가 저장된다. 실제 미디 데이터는 delta-time 과 이벤트(Event)가 한 쌍이 된 집합으로 구성된다.

delta-time 은 항상 이벤트 앞에 존재한다. delta-time 은 현재 이벤트와 바로 앞에 위치한 이벤트와의 시간 간격으로 그 단위의 기준은 헤더 청크에서 설정한 4 분음표 길이에서 비롯되며 가변길이를 가진다. 만약 이벤트가 동시에 발생한 경우에는 값이 0 이 된다.

이벤트는 MIDI 이벤트와 Sysex 이벤트(System Exclusive Event), 그리고 메타 이벤트(Meta Event)로 구분할 수 있다. MIDI 이벤트는 주로 음표의 시작과 끝, 음의 성격 등을 나타내는 음표 관련 이벤트이고 Sysex 이벤트는 MIDI System Exclusive Message 를 주로 저장하기 위한 이벤트이다. 메타 이벤트는 박자, 빠르기, 조 바뀜과 가사 등, MIDI 이벤트와 Sysex 이벤트에 포함되지 않는 기타 이벤트를 포괄하고 있다. 모든 이벤트들은 상태 바이트(Status Byte)와 이벤트 데이터로 구성되며 같은 상태 바이트를 가지는 이벤트가 반복될 때에는 뒤에 오는 이벤트의 상태 바이트는 표시되지 않는데 이것을 러닝 스테이터스(Running Status)라고 한다.

다음은 이상에서 기술한 트랙 청크의 구조를 나타내고 있다.

```
[Syntax of a Track Chunk] ;
<chunk type> <length>
<delta-time> <status> <event data>
<delta-time> <status> <event data>
        :
```

2.2 미디 파일에서의 Chord 정보 유지

사람들이 악보를 볼 때 단순히 오선 위에 음표만 있는 것보다 현재 음의 성격을 나타내는 코드(Chord)가 함께 있는 것이 그 곡을 이해하는 데에 더욱 효과적이다. 따라서 악보에 코드를 출력하기 위하여 코드 정보를 미디 파일에 저장하는 방안에 대해 기술한다.

음표의 출력에 직접적인 관계가 없는 이벤트인 메타 이벤트에서는 코드 정보를 기록하기 위해 특별한 준비를 하지 않고 있다. 따라서 현재로서는 직접적으로 코드 정보를 기록할 수 없기 때문에 다른 잘 사용되지 않는 영역을 대신 사용하는 수밖에 없다. 이에 적합한 영역으로 텍스트를 저장하도록 준비된 곳이 있다. 이 영역은 일반 문자열을 자유롭게 저장할 수 있는 곳으로 코드 정보를 코드 표시 그대로 문자열로서 저장한다.

3. 미디 파일을 이용한 악보 생성

3.1 이벤트 리스트의 구성

미디 파일에 기록된 이벤트들은 그 성격에 따라 저장된 것이 아니라 발생한 순서에 따라 저장되어 있어서 파일을 읽을 때 이들을 성격대로 분류하여 따로 유지할 필요가 있다. 이벤트의 성격대로 분류될 것은 음표를 비롯하여 빠르기, 조 바뀜, 박자, 가사, 코드이다. 이 이벤트들을 읽어서 유지하는 방법으로는 이벤트가 발생한 순서에 따라 저장되어 있다는 것을 고려해 볼

때, 리스트(List) 구조가 가장 적합하다.

따라서 미디 파일을 읽고 분석하는 과정을 통해 음표 리스트, 빠르기 리스트, 조 바뀜 리스트, 박자 리스트, 가사 리스트, 코드 리스트가 구성된 결과가 나와야 한다.

다음의 코드(Code)는 각 리스트의 구조를 나타낸 것으로 모든 클래스는 클래스 CEvnet 로부터 파생된다.

```
class CEvent
    unsigned int OccurTime;  // event 의 발생 시간

class CNoteEvent : public CEvent          // 음표 event
    unsigned char Channel;
    unsigned char NoteNo;    // 음 번호
    unsigned char Velocity;  // 음량
    int DelayedTime;         // 음의 길이

class CTempoEvent : public CEvent         // Tempo event
    unsigned short Tempo;

class CKeySigEvent : public CEvent        // 조바뀜 event
    unsigned short KeySig;

class CTimeSigEvent : public CEvent       // 박자 event
    unsigned short TimeSig;

class CLyricEvent : public CEvent         // 가사 event
    char *Lyric;

class CChordEvent : public CEvent         // Chord event
    unsigned short Chord;
```

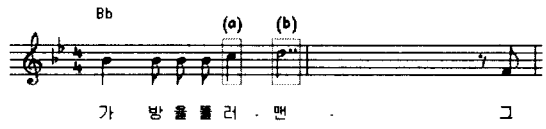
3.2 이벤트 리스트를 통한 악보 생성

3.2.1 음표의 정규화

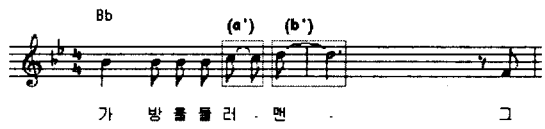
악보상의 음표 표기방식은 일정한 규칙이 있으며 사람이 악보를 볼 때 쉽고 정확히 볼 수 있도록 구성되어야 한다. 잘못 표기되지는 않았지만 읽기에 어렵도록 되어 있는 악보라면 보는 사람으로 하여금 불편함을 느끼게 한다. 그러므로 악보를 정확히 표기하는 것 뿐만 아니라 보기 편하도록 표기하는 것도 매우 중요하다.

음표 리스트에 저장된 음표 이벤트는 마디의 길이나 음표의 나타내어질 모습 등이 고려되지 않은 채 음표의 시작 위치와 음표의 길이만을 가지고 있다. 이것을 그대로 표기한다면 정확한 악보 표현이 어려워지게 되고 더 나아가 실제 곡과는 다르게 되어 잘못된 악보가 된다. 따라서 악보를 생성하기에 앞서 미디 파일의 분석을 통해 얻어진 각 이벤트의 리스트들 중 음표 리스트를 정규 형식으로 변환해야 한다.

다음의 두 그림은 정규 형식으로 변환하지 않은 채 악보를 생성한 것과 정규 형식으로 변환한 후 생성된 악보를 비교하고 있다. <그림 1>의 (a)는 잘못 표기되지는 않았지만 악보를 읽기에 불편한 형태이고 (b)는 아예 잘못 표기된 형태이다. 이들은 모두 <그림 2>의 (a')과 (b')의 형태와 같이 변환되어 출력된다.



<그림 1> 변환하지 않고 악보를 생성한 경우



<그림 2> 정규 형식으로 변환한 후 악보를 생성한 경우

3.2.2 음표와 가사, 음악 기호의 입력

하나의 악보는 수많은 마디가 순서대로 결합되어 만들어진다. 하나의 마디는 악보를 독립적인 가장 작은 형태로 나누었을 때 기본 단위라고 할 수 있다.

전체 악보를 생성하는 과정에서 모든 음표들은 한꺼번에 일률적으로 악보에 올라가기 보다는 각기 자신들이 속해 있는 마디 단위로 나뉘어 마디별로 그려진 후 개별적으로 생성된 이 마디들을 하나로 합치는 것이 더욱 효율적이다. 정규 형식으로 변환되어 리스트 형태로 유지된 음표 이벤트들은 마디별로 쉽게 분류될 수 있다.

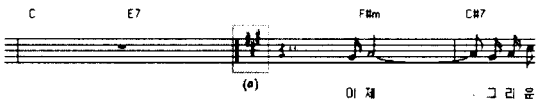
음표를 마디에 그릴 때에는 미리 음표의 길이와 높낮이를 파악하고 이에 맞는 이미지를 준비하여 해당되는 위치에 삽입한다. 이 때 하나의 마디에 포함된 음표 이벤트가 너무 많을 경우 음표를 삽입할 공간이 부족하므로 미리 음표의 수를 파악하고 적당한 마디의 길이를 계산하여 음표가 삽입될 충분한 간격을 확보한다.

가사는 음표와 함께 입력한다. 각 가사 이벤트

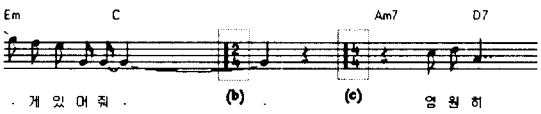
들은 모두 음표 이벤트들과 일대 일 대응되므로 해당 음표의 바로 아래 부분에 위치시키며 음표에 대응되는 가사가 없다면 하이픈이나 다른 대체 문자로 출력한다.

곡의 중간에 박자나 조가 바뀌는 이벤트가 발생할 경우 악보에서는 이에 적절한 처리를 해 주어야 한다. 이와 같은 이벤트들은 마디의 중간에 발생하지 않고 항상 마디의 처음 부분에서 일어난다. 따라서 각 마디별로 음표를 생성하기 전에 박자나 조가 바뀌었는지를 조사하여 이에 해당하는 기호를 삽입한다.

다음 <그림 3>은 조 바뀜 이벤트가 발생한 위치에 그 표시를 한 것으로 (a) 부분에서 조표가 다장조에서 가장조로 바뀌고 있음을 알 수 있다. <그림 4>는 박자 이벤트가 발생하여 변박이 된 것을 표시한 것으로 (b)에서 원래 4/4 박자의 곡이 2/4 박자로 바뀌었다가 다시 (c)에서 원래의 박자인 4/4 박자로 바뀌는 모습이다.



<그림 3> 조 바뀜의 처리 예



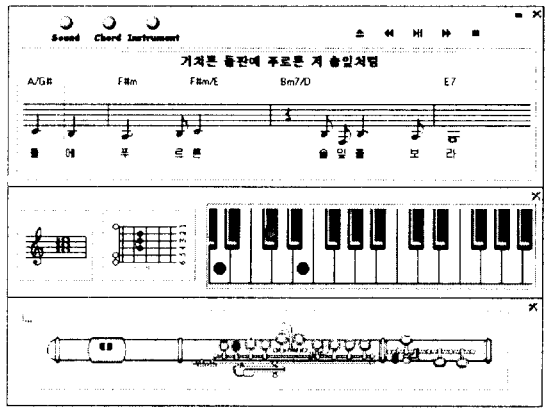
<그림 4> 박자 바뀜의 처리 예

3.2.3 기타 처리해야 할 사항

악보의 윗부분에는 앞서 밝힌 바와 같이 코드를 출력함으로서 악보를 보는 이가 곡의 구성을 더욱 잘 이해할 수 있도록 한다. 코드의 삽입은 음표를 삽입하는 방법과 마찬가지로 각 마디의 생성시 해당 마디에 포함되는 코드를 이벤트 발생 시점에 따라 그 위치를 계산하여 입력한다.

마디별로 생성된 악보를 하나로 합친 후 이음줄과 붙임줄을 그림으로 악보 생성의 모든 과정이 끝난다. 이음줄과 붙임줄은 음표의 정규적인 형식으로 변환하는 처리를 인해 나누어진 음표를 이어주는 역할을 한다. 이들의 출력 여부는 모든 음표 이벤트를 비교하여 결정되고 그려지는 이음줄과 붙임줄은 음표와 음표사이에 일정한 높이의 호 모양으로 그려지게 된다.

다음 <그림 5>는 이상에서 기술한 방법으로 생성한 악보를 이용하여 악기 및 노래를 연습할 수 있도록 하는 프로그램이다.



<그림 5> 악기 및 노래 연습 프로그램의 실행

4. 결론

이상에서 미디 파일의 한 형식인 SMF 형식으로 미디 자료가 저장되는 구조를 논의하였고 이 형식으로 저장된 미디 정보를 통하여 악보를 생성하는 방법에 대해 기술하였다.

이 방법으로 생성된 악보는 악기 및 노래 연습을 위한 프로그램에 사용되어 연습에 많은 도움을 줄 것으로 기대된다.

[참고 문헌]

[1] 김태희, 컴퓨터 음악의 세계 MIDI 기초개론, 정보게이트, 1999

[2] 박훈, 포크기타 첫걸음, 음악도서 삼호출판사, 1977

[3] 오상문, 볼랜드 C++ Builder 완벽 가이드, 영진출판사, 1998

[4] 이상엽, Visual C++ Programming Bible Ver 5.x, 영진출판사, 1997

[5] 이정선, 새 이정선 기타교실, 도서출판 이정선 음악사, 1996

[6] 정우철, Borland C++ Builder 4 Programming Bible, 정보문화사, 1999

[7] 정우철, 알기쉬운 볼랜드 C++ 빌더 3, 정보문화사, 1998