구조로 알아보는 BMS 포맷의 역사와 발전방향

**I. 개요**

오락실에서 즐기는 게임을 PC로도 즐기려는 요구는 계속되어왔다. 음악의 연주를 주제로 하는 리듬게임도 예외가 아니다. 게임 개발사에서 공식적으로 PC판을 지원하지 않아도, 유저들 스스로 아케이드와 비슷한 환경을 PC로도 구축하려는 노력이 있어왔다. 그 결과물은 리듬게임의 종류마다 다르겠지만, 최초의 아케이드 리듬게임인 beatmania를 PC로도 연습하기 위해 개발된 BMS 포맷의 구조와 역사를 리듬게임의 발전에 초점을 맞추어 이 글에서 소개하고자 한다. 더 나아가 롱노트 기술에 초점을 맞춘 BMS 포맷 표준화 논쟁과 그 한계 및 개선방안을 제시하고자 한다.

**II. BMS**의 역사와 구조

**1. BMS**의 등장

**1)** beatmania의 발매와 BMS의 탄생

1998년, 일본에서 최초의 아케이드 리듬게임인 beatmania가 발매되었다. 5개의 건반형 버튼과 1개의 스크래치형 입력장치로, 낙하하는 노트가 특정 높이(판정선)에 도달할 때 입력하여 음악을 연주하는 (적어도 연주하는 느낌을 내는) 게임이다. 음악은 키 입력을 받지 않아도 자동으로 재생되는 배경음 영역과, 노트의 의도한 타이밍에 맞추어 입력을 받아야 재생되는 키음(key音) 영역으로 구성되는데, 배경음과 그에 맞춰 정확한 타이밍에 재생되는 키음들이 하나의 음악을 이루는 것이다. (배경음과 키음을 구분하지 않고 사용된 음악 파일을 모두 키음으로 부르기도 하며, 이 글에서도 상황에 따라 그렇게 사용한다.)

BMS는 Be-Music Script의 줄임말로 기본적으로 beatmania를 PC로 연습하기 위해 고안된 포맷이며, 1998년 BMS 구동 프리웨어 BM98이 개발되면서 .bms의 확장자로 처음 모습을 보였다. (Be-Music은 beatmania를 패러디한 이름이다) BMS 파일 자체는 기본적인 텍스트 에디터로도 어렵지 않게 편집할 수 있으며, 구조 또한 어렵지 않다. 이러한 특징으로 명령어 추가 등의 확장이 쉽게 이루어졌고, 이러한 확장은 출시되는 아케이드 건반형 리듬게임의 기능을 도입하는 경향이 있다.

**2)** BMS의 기본적 구조

BMS 파일의 구조를 추상화하면 한 방향으로 계속 읽어가는 길고 넓은 천공카드 여러 개의 (물론 읽는 방향으로의) 연결로 볼 수 있다. 천공 카드의 개수와 각각의 길이와 구멍의 위치, 읽어나가는 속도도 자유롭게 조절할 수 있으나(정지도 가능하다), 역방향으로는 읽을 수 없다. (또는 그런 시도를 하면 의미 없는 파일이 된다) 실제 BMS 파일은 각 구멍마다 다른 음악 파일을 재생시키도록 할 수 있으므로 엄밀히는 3차원으로 확장시킨 천공 카드 묶음이 된다.

이하는 BMS 파일에서 부가적인 기능은 배제한 채, 가장 기초적인 기능인 음악 파일의 할당과 배치가 어떻게 이루어지는지에 대해 다룬다.

음악 파일의 할당은 아래와 같은 구문으로 한다.

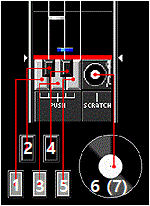
#WAV01 foo.wav

위와 같이 00을 제외한 01-FF의 이름을 가진 255개 중 하나의 슬롯(붉은 글씨)에 음악 파일(초록 글씨)을 할당할 수 있다. 음악 파일은 bms 파일과 같은 폴더 내에 위치해야 한다.

음악 파일의 배치는 아래와 같은 구문으로 한다.

#00714:00010000

그림 1 beatmania의 조작부와 BMS 파일에서 대응되는 건반 번호

위와 같이 할당한 음악 파일을 각 영역에 의도한 타이밍에 재생되거나 연주할 수 있게 한다. 먼저 마디(bar) 번호(붉은 글씨)를 지정한다. 마디 번호는 000부터 시작하여 1씩 증가한다. 그 다음 PLAYER의 번호(초록 글씨)를 지정한다. 여기서는 음악 파일을 배경음 영역에 할당하는 경우인 0과 키음 영역에 할당하는 *일반적인* 경우인 1 (양측의 입력장치를 모두 사용하는 더블 플레이의 경우는 1과 2)의 상황만 고려한다. 그리고 연주 시 입력해야 할 키 번호(파란 글씨)를 지정한다. 배경음인 경우 1을, 키음인 경우 beatmania의 입력 장치와 그림 1과 같이 대응한다.

6은 일반적인 스크래치 입력에 지정된다. 7은 초기 beatmania 시리즈에 존재하였던 스크래치 FREE ZONE 입력에 지정되며 현재는 쓰이지 않는다.

타이밍은 콜론 이후의 문구(보라색 글씨)에 의해 지정된다. 이 부분을 두 글자 단위(노트)로 나눈다. (00 01 00 00) 각각의 노트가 그 슬롯에 할당한 음악 파일을 의미하며 00은 아무 파일도 재생하지 않음을 뜻한다. 마디를 00을 포함한 노트의 개수로 분할한다. 마디(007)의 시작 위치를 0, 마디의 끝의 위치를 1이라 하면, 각 음악 파일을 재생시키는 노트는 0/n, 1/n, …, (n-1)/n의 위치에 놓이게 된다.

위의 두 구문을 입력한 경우 기본값인 BPM 120 (1분에 4분음표 120개) 과 마디 박자를 4/4로 읽는다면 곡 시작 14.5초 후에 슬롯 01에 할당된 키음 foo.wav를 4번 건반을 눌러 연주해야 한다.

배경음의 경우 같은 마디 번호(XXX)에 대해 #XXX01: 로 시작하는 구문을 ‘많이’ 사용할 수 있다. (최대로 인식할 수 있는 수는 구동기마다 다르다) 그러나 키음의 경우 배경음처럼 중복하여 할당할 수 없으며, #XXX1Y: 를 여러 번 입력한 경우 BMS 파일의 끝과 가장 가까운 구문을 처리한다.

**2. BME로의 확장**

BME는 .bme의 확장자를 가지며 마지막 글자인 E는 extended를 뜻한다. 그렇다면 무엇을 확장(Extend)한 것이고, 왜 확장할 필요가 생겼을까.

*일반적인* 답은 건반의 개수 확장이다. 1999년, 일본에서 beatmania IIDX가 발매되었다. beatmania IIDX 와 달리 7개의 건반형 버튼, 1개의 스크래치형 입력장치를 사용하였다. 당연히 이에 맞춰 BMS 포맷을 확장하려는 움직임이 있었다. (왼쪽에서부터 순서대로 버튼 번호를 붙일 시) 1-5번 건반을 순서대로 1-5번에 할당한 것처럼, 추가된 6-7번 건반을 6-7에 할당하는 것이 가장 직관적이었으나, 6과 7은 이미 스크래치에 할당되어서 이는 어렵게 되었다.

초기에는 5키의 더블 플레이(1P, 2P 입력장치를 모두 사용한 패턴)를 7키 싱글처럼 활용하자는 Project2DX가 고안되었다. 즉, 11-15까지는 원래의 5키처럼 1-5번 건반에, 새로운 6-7번 건반은 21-22에 할당하는 것이다. 그러나 이는 곧 5키의 더블 플레이 패턴을 7키의 더블 플레이로 확장하기가 어렵고, 5키 더블 플레이와 7키 싱글 플레이 패턴을 구동기 측에서 구분하기 번거로운 문제가 발생하였다.

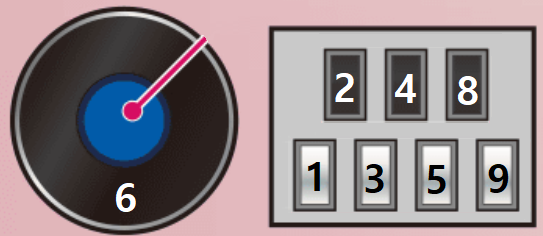
따라서 추가된 두 개의 건반은 플레이어를 정의하는 1을 유지한 채, 키 번호 각각 8과 9에 할당하는 것이 BME 파일의 표준이 되었다. 더블 플레이 패턴으로의 확장이 쉬우며, 5키 BMS 파일과의 하위호환이 됨이 강점이다. 다만 이는 5키 BMS 파일과 달리 키 번호가 직관적이지 않은 문제점이 존재한다. (그림 2 참고) 사견이지만, BME 파일의 표준을 제안할 때, BMS 파일과의 하위호환을 포기하고 (확장자로 구분되므로) 스크래치를 0에, 6-7번 건반을 6-7 할당하는 것이 맞지 않았나 한다.

그림 2 beatmania IIDX의 조작부와 BME 파일에서 대응되는 건반 번호

그래픽 인터페이스를 제공하는 BMS 편집 프로그램이 있기 때문에 위와 같은 문제는 현재 크게 체감할 수 없으나, 편집 프로그램에서 지원하지 않는 명령어를 추가하기 위해 텍스트 형태로 편집하는 경우나, 구동기를 개발할 경우 그 단계가 조금 더 복잡하게 되는 문제는 여전히 존재한다.

엄밀하게 BME는 가장 초기의 BMS 파일이 지원하지 않는 명령어들이 추가된 모든 파일을 의미한다. 따라서 5키 파일이어도 그래픽적, 혹은 다른 게임 시스템적으로 확장된 명령어를 포함한다면 그 확장자가 .bme일 것이다. BMS와 BME의 전체적인 구조가 동일하지만 확장자가 변경된 이유도 구세대 구동기가 BME 파일을 인식하지 못하게 함이 크다. 다만, 현재 사용되는 대부분의 BMS 구동기는 확장자를 신뢰하지 않아 (즉, .bme 파일의 확장자를 .bms로 변경해도 아무 문제가 없다.) 이러한 구분이 무의미해졌다. 따라서 현재의 .bme의 확장자는, 제작자들이 7키를 기반으로 만든 파일에 관습상 부여하는 의미가 크다.

**3. BML의 도입**

**1)** EZ2DJ의 발매와 롱노트

1999년, 한국에서는 아케이드 리듬게임 EZ2DJ The 1st TRACKS가 발매되었다. beatmania와 매우 유사한 입력장치와 게임 방법을 가지고 있었으나, 페달을 입력장치에 추가하였으며 롱노트 시스템을 도입하여 beatmania와는 차별화되었다. 이러한 요소를 BMS 파일에 도입하려는 움직임 또한 당연히 생겼다. 페달은 쓰이지 않게 된 키 번호 7에 할당하려는 FPP (Foot Pedal Package) 등의 시도가 있었으나 사장되었다. 다만, 롱노트는 현재까지 활발하게 사용되는 요소로, 근 10년간 개발되어 사용되는 모든 구동기가 지원한다고 봐도 무방하다.

그림 3 EZ2AC에서의 롱노트

롱노트는 길이가 긴 음악 파일의 연주감을 살리려는 목적으로 사용되며, EZ2DJ의 게임 화면에서도 일반 노트보다 길게 표시된다. 롱노트의 처음 끝을 일반 노트처럼 누르는 대신, 마지막 끝이 판정선을 통과할 때까지 유지한다. 입력을 멈추는 판정은 EZ2DJ 시리즈 내에서도 일관적이 못하였는데, 초기 작품에서는 마지막 끝이 판정선을 통과하는 타이밍에 맞추어 입력을 중지해야 했지만, 2000년 출시된 후속작 EZ2DJ 2ndTRAX에서는 의무적으로 중지해야 할 필요가 사라졌고 지금까지 이어지고 있다.

이러한 판정 체계의 혼란처럼 BML의 도입은 시작부터 순탄치 않았다. BMS, BME의 하위호환을 유지하면서 롱노트를 어떻게 작성할 것인지, 긴 음악 파일의 연주감을 살리기 위해서는 작성된 롱노트를 구동기 측에서 어떻게 처리해야 되는지에 대해 의견이 통일되지 않았기 때문이다.

**2)** 롱노트란 무엇인가

실제로도 롱노트는 처음 끝과 마지막 끝의 존재, 시각적으로 유사한 생김새 외에는 아케이드 리듬게임마다 처리와 판정 방식이 모두 다르다. 마지막 끝에서 의무적으로 떼야 되는 지의 유무, 계속적인 입력을 짧은 시간 중지하는 것을 허용할 지의 유무, 롱노트를 일반 노트 몇 개의 비중으로 처리할 것인지의 유무 등이 모두 다르며 이외의 세부적인 차이점도 매우 많다. 심지어 처음 끝에 맞추어 의무적으로 입력할 필요 없이 미리 누르고 있어도 되는 리듬게임도 존재한다. 이러한 혼란은 다른 리듬게임 개발사 사이에서 롱노트 요소를 표준화할 이유가 전혀 없기에 생겨왔다.

그러나 BMS는 다르다. 롱노트를 기술하는 방법이 시판된 아케이드 리듬게임 수만큼 많아진다면 BMS를 편집하는 입장에서도, 구동기를 개발하는 입장에서도 고역이다. 특히 구동기는 동일한 롱노트 구문을 읽을 시 위의 요소들이 모두 같게 적용되어야 하는 제한이 있다. (하나의 구문을 다르게 처리할 수는 없다!) 구동기는 각 리듬게임 시스템에 맞춰 여러 개가 개발되더라도, 롱노트 기술은 하나의 포맷으로 통일하려는 시도가 있어왔지만 결국 실패하였다.

그러나 흥미로운 것은 구동기의 표준이 실질적으로 정해진 점이다. BMS 포맷이 고안된 지 20년이 넘어가는 현재, 여러 확장 명령어와 이를 지원하는 수많은 구동기가 개발되었으나, 결국 2010년 마지막으로 업데이트된 Lunatic Rave 2가 표준이 되었다. 그 이유는 beatmania IIDX와 상당한 유사한 시스템을 가지며, 인터넷 랭킹을 지원하기 때문이다. 따라서 이하의 명령어는 Lunatic Rave 2와, 가끔씩 입문용 BMS 구동기로 추천되는 ruv-it!이 지원하는 한도 내에서 설명한다. (롱노트 도입 초기에 롱노트 기술을 위한 여러 확장 명령들이 제안되었으나 유의미한 것만 설명한다는 것이다. BMS 포맷의 특성상 공식적인 표준은 없지만, 이 글에서의 표준은 유의미하게 많이 사용된다는 의미와 같다.)

**3)** 롱노트 표준화 논쟁

**가)** 롱노트의 기술 방법

**(1)** MGQ 포맷

MGQ 포맷은 BMS에 롱노트를 도입하기 위해 2001년 제안되었다. 롱노트 기술 영역이 일반 노트와 구분되게 PLAYER의 번호로 5(더블 플레이는 5와 6)를 사용한다. 아래와 같은 예시가 있다.

#00853:01010100

이 영역에 쓰인 노트는 처음 쓰인 위치에서 롱노트의 처음 끝이 되며, 연속해 쓰인 만큼 길이를 가지고, 이후 00이 쓰인 위치에서 롱노트의 마지막 끝이 된다. 위의 예시는 8번째 마디가 시작할 때 3번키 영역에 foo.wav를 재생하는 롱노트의 처음 끝이 나오며, 8번째 마디의 3/4 지점에서 롱노트의 마지막 끝이 나오게 하는 명령이다.

그러나 Lunatic Rave 2는 이 방법을 지원하지 않으며, ruv-it!은 지원하나 필수적으로 BMS 파일 내에 #LNTYPE 2를 기술해야 한다. 롱노트 작성이 번거롭고 비직관적인 문제가 존재하여 현재는 사장되었으나, 최초로 널리 알려진 BMS 상의 롱노트 기술법이라는 의의가 있다.

**(2)** RDM 포맷

RDM 포맷은 MGQ의 위와 같은 문제점을 개선한 롱노트 기술 방법이다. MGQ와 유사하지만 롱노트의 마지막 끝이 00 이외의 노트가 처음 쓰이는 위치가 되는 차이점이 있다. 즉, MGQ로 기술된 위 예시를 RDM으로 다시 기술하면 아래와 같다.

#00853:01000100

Lunatic Rave 2와 ruv-it! 모두 RDM 포맷을 지원한다. RDM 도입 초기에는 필수적으로 BMS 파일 내에 #LNTYPE 1을 기술해야 했으나, 현재는 이 방법이 하나의 표준이 되어 그럴 필요가 없다.

**(3)** RDM TYPE 2 포맷

RDM TYPE 2 포맷은 현재 롱노트를 기술하기 위한 또다른 표준이다. 일반 노트와 동일한 PLAYER 번호를 사용하는 것이 강점으로, 롱노트의 마지막 끝을 정의하기 위해서만 사용되는 노트 번호를 #LNOBJ FF 형태로 BMS 파일 내에 기술해야 한다. #LNOBJ 명령으로 지정된 노트를 찾는다면, 이를 롱노트의 마지막 끝으로, 같은 키 번호의 그 앞 노트를 롱노트의 처음 끝으로 하여 롱노트로 처리한다. 따라서 위 예시를 RDM TYPE 2로 다시 기술하면 아래와 같다.

#LNOBJ FF  
#00853:0100FF00

Lunatic Rave 2와 ruv-it! 모두 RDM TYPE 2 포맷을 지원한다. BML 포맷은 엄밀하게는 이 방법만을 의미하나, 현재의 BML은 BME에서 언급한 것과 같은 이유로 제작자가 롱노트를 이용한 BMS 데이터 작성시 관습적으로 지정하는 확장자(.bml)로 기능한다.

**나)** 롱노트의 처리방법

롱노트는 리듬게임마다 매우 다른 특징을 가진다 위에서 언급하였다. 이하는 그 특징이 어떻게 다른지와 Lunatic Rave 2와 ruv-it!에서 롱노트가 어떻게 처리되는지에 대해 논의한다. BMS 포맷 자체보다는 구동기에 대한 논의이지만, 실질적으로 위의 구동기가 표준으로 자리잡은 현재 의미 있는 논의가 될 것이다.

**(1)** 롱노트의 틱

롱노트는 일반 노트 몇 개의 비중(점수나 판정 등의 측면에서)으로 인식되어야 하는가? RDM (TYPE 2) 포맷을 고려하면은 가장 직관적인 답은 처음 끝과 마지막 끝 2개가 된다. 하지만 롱노트를 처음 도입한 EZ2DJ 시리즈의 경우 2007년 이전까지는 일반노트 한 개와 같은 비중으로 처리하였다. 반면 2001년 한국에서 출시된 아케이드 댄스 리듬게임인 펌프 잇 업 EXTRA는 롱노트의 입력 여부를 의도한 간격(2분박이나 4분박 등)마다 체크하는 ‘틱’ (tick) 시스템을 선보였다. 시스템 특성상 롱노트의 비중은 롱노트의 길이에 비례하게 되었으며, 롱노트를 체크하는 타이밍 사이에는 입력을 중지하여도 문제가 없다.

하지만 이는 댄스 리듬게임이기에 건반형 리듬게임을 기반으로 하는 BMS와는 관계가 없는 것으로 보였다. 따라서 대부분의 BMS 구동기는 롱노트의 비중을 (EZ2DJ와 같은) 일반 노트 1개 또는 (직관적으로) 2개로 처리하였다. 그런데 2007년 출시된 EZ2DJ 7thTRAX에서 위와 같은 롱노트 틱 시스템을 신곡에 적용하였다. 이마저도 점수 시스템의 혼란을 고려해 기존 곡들은 롱노트가 일반 노트 1개의 비중을 유지하는 기이한 모습을 보였다. 같은 건반 게임 내부에서도 롱노트 처리를 다르게 하니 게임 유저는 물론 BMS 구동기 개발 측에서도 혼선을 겪을 수밖에 없었다.

이러한 혼란은 Lunatic Rave 2와 ruv-it!에서도 그 모습을 찾을 수 있다. Lunatic Rave 2는 롱노트를 일반 노트 1개의 비중으로 처리하나, RDM 포맷으로 작성된 BMS 파일을 스캔 시 롱노트의 개수만큼 총 노트수가 증가하여 계산되는 버그가 있다. (이는 롱노트의 비중을 2개로 계산하는 것이 처리가 쉽다는 반증이기도 하다.) ruv-it!에서는 아예 설정에서 틱 시스템의 적용 유무를 선택할 수 있으며 기본값은 적용 상태이다. 적용시 16비트(일반적으로 4분박)의 틱 시스템을 가지며, 미적용시 Lunatic Rave 2와 같이 1개의 비중을 가진다.

**(2)** 롱노트의 끝

롱노트의 처음 끝과 마지막 끝에 정확히 입력을 시작하고 중지해야 하는가? 대부분의 건반형 리듬게임은 롱노트의 처음 끝에 맞추어 정확히 입력을 시작해야 한다. 그렇지 않으면 노트를 의도한 타이밍에 입력하여 키음을 재생시키는 연주감이 퇴색되기 때문이다. (이런 경우 롱노트로만 구성된 패턴을 재생한다면 모든 키를 음악 시작부터 끝까지 단순히 계속 누르는 것으로도 만점을 받을 수 있을 것이다. 실제로 현재 펌프 잇 업 시리즈가 이러며, 게임 내의 ‘미운오리새끼’라는 곡에 롱노트로만 구성된 패턴도 존재한다.)

그러나 롱노트의 마지막 끝에 대해서는 의견이 갈린다. 연주감이 의무적인 입력 중지 형태로도 실현되는가에 대한 입장이 유저마다 달랐기 때문이다. 롱노트를 최초로 도입한 EZ2DJ 시리즈도 초기에는 이에 대해 일관적이지 못함으로 볼 때 이는 확답을 내리기 어려움이 분명하다. 현재의 건반형 리듬게임은 롱노트의 비중이 일반노트 1개이거나 틱 시스템을 도입한 경우 마지막 끝에서 떼지 않아도 되며, 롱노트의 비중이 일반 노트 2개인 경우 마지막 끝에서 의무적으로 떼야 한다.

Lunatic Rave 2와 ruv-it!에서도 이러한 특징이 다르게 반영된다. 모두 롱노트의 처음 끝에 정확히 입력을 시작해야 한다. Lunatic Rave 2는 마지막에 뗄 필요 없이 처음 끝에 입력하는 것만으로 정확도를 판단한다. ruv-it!은 마지막에 떼지 않는다고 실패 판정을 내리지는 않지만 처음 끝 입력 시의 정확도에 비해 롱노트에 대한 종합적인 정확도가 감소하게 된다.

**(3)** 스크래치의 롱노트

이 부분은 BMS 구동기에서의 표준 논쟁보다는 BMS 제작 및 플레이 환경에서의 경향을 논한다.

스크래치형 입력장치에서 롱노트는 어떻게 실현되는가? 스크래치의 롱노트(이하 롱스크래치)는 1999년 EZ2DJ The 1st TRACKS Special Edition에서 처음 등장하였다. 이 게임에서 선보인 처리 방법은 롱스크래치의 처음 끝에 맞추어 스크래치의 회전을 시작하고 이 회전 방향대로 회전을 롱스크래치의 마지막 끝까지 유지하면 되었다.

그러나 입력장치가 키보드로 제한된 BMS 환경에서는 롱스크래치의 아이디어가 황당할 뿐이었다. 키보드의 특정 키를 스크래치처럼 회전시킬 수 있는가? 결국 스크래치 또한 일반적인 키에 할당되고, 처리 방법 또한 일반적인 롱노트와 같게 될 수밖에 없었다. 이걸로 마무리되었다면 이 부분을 서술하는 의미가 없을 것이다.

그런데 2009년 출시된 beatmania IIDX 17 SIRIUS (이하 IIDX)에서 롱스크래치를 백 스핀 스크래치라는 이름으로 도입하였다. 처리방법은 EZ2DJ의 롱스크래치와 유사하나, 롱스크래치의 마지막 끝에 맞춰 반대 방향으로 회전시켜야 한다. 이는 독특한 연주감을 선보여서 beatmania IIDX 최신작까지 많이 사용되고 있다.

그러나 이를 BMS 환경으로 이식하는 것이 문제였다. Lunatic Rave 2는 IIDX과 유사한 시스템을 강점으로 가졌으나, 도저히 백 스핀 스크래치만큼은 키보드 환경에서 구현할 수 없던 것이었다. 키보드의 특정 키를 누르다가 반대 방향으로 누르는 개념이 타당한가?

유의할 점은 2009년은 BMS 초창기와 다르게 키보드 플레이가 대다수를 차지하지 않는다는 것이다. IIDX의 입력장치를 본뜬 콘트롤러가 매니아 유저층 사이에서 보급된 때이며, 약간 과장하자면 BMS는 이들 사이에서 IIDX 연습 그 이상의 목적은 없었다. (달리 생각하면 개발 초기의 순수한 목적으로 돌아간 것이 아닌가 싶다.) Lunatic Rave 2에서 그들은 IIDX의 신요소인 백 스핀 스크래치를 연습하고자 하였고, 이는 BMS 제작자들 사이에서 롱스크래치 구현 방식까지 변화하게 하였다.

이전에는 롱스크래치 구현을 일반적인 롱노트와 같이 하였고 별다른 특이사항은 없었다. 그러나 백 스핀 스크래치 등장 이후 롱스크래치의 마지막 끝을 조금 앞으로 (제작자에 따라 다르지만 대체로 25-100ms) 이동시키고 원래 마지막 끝의 위치에 일반 스크래치 노트를 하나 추가하는 형태로 구현하는 것이 대세가 되었다. 스크래치의 입력은 정지된 상태에서 회전시키거나, 회전하고 있는 방향의 반대 방향으로 회전시키는 것으로 이루어짐과 Lunatic Rave 2의 롱노트 판정 시스템을 고려할 때 백 스핀 스크래치를 상당히 유사하게 구현한 것이다. 이는 건반 게임이 도입한 새로운 시스템과 BMS 플레이 및 제작 환경이 만들어낸 실질적인 표준인 셈이다.

그러나 이 경향에 대한 반발은 여전히 존재한다. 먼저 스크래치에서 일반노트와 롱노트를 같은 마디에서 여러 개를 기술하면 플레이 시 가독성이 매우 떨어진다. 특히 롱노트의 끝을 강조하는 스킨을 사용할수록 이러한 문제가 두드러진다. 또한 키보드를 입력장치로 사용하는 경우 이러한 롱스크래치는 조작감이 독특하다 못해 불편하다. 롱스크래치 마지막 끝 바로 뒤에 있는 일반 노트를 입력하기 위해서는 입력을 잠시 중단하는 것이 불가피한데, 일찍 중단하면 롱노트의 마지막 끝보다 이르게 떼일 위험이 있으며, 늦게 중단하면 일반 노트를 의도한 타이밍에 입력할 수 없다. 개인적으로도 이러한 경향성은 키보드 유저를 배려하지 않는 것이라 생각하지만 한 번 자리잡은 대세는 어찌할 수 없다는 생각을 들게 한다.

**III. 발전방향 제안**

BMS는 유저가 직접 음악 파일과 패턴 등을 편집할 수 있는 자유도가 큰 강점인 포맷이다. 그러나 (Lunatic Rave 2로 구동 시) 정말로 자유로운 포맷인가? 큰 불편을 느낄 요소는 없지만 생각보다 미묘한 제한이 존재한다. 그 제한으로 비롯되는 문제점을 서술하고, 개인적으로 생각하는 포맷 또는 구동기 단위의 해결방법을 제안하고자 한다.

**1. 키음 슬롯의 제한**

현재 키음 슬롯은 2자리의 36진법 문자로 (01~ZZ) 1295개로 그 수가 제한되어 있다. BMS 개발 초창기의 255개 (01~FF) 보다는 대폭 증가한 것이나, 이것도 부족하다는 의견이 있다. 초창기부터 연주감을 살리기 위해 BMS 제작 측에서 키음 데이터를 제작할 때 가상악기 트랙 하나를 통째로 녹음한 뒤 음표 단위로 분리하는 사례가 많았고, 최근에는 실제 악기를 연주 후 녹음해 음표 단위로 분리해서 키음에 사용하는 경우도 있다. 이러한 시도를 세 개의 악기만 BMS에 적용하려 해도 1295개의 슬롯은 부족하다.

이를 해결하기 위해 BMSON 포맷은 음악을 트랙별로 분리하여 녹음만 하면 트랙 파일을 BMSON 파일 내에서 음표 단위로 분리할 수 있게 하였다. 그러나 기존 BMS 편집기나 구동기에서 BMSON을 인식 후 처리하게 하려면 기존의 BMS 처리 과정과는 완전히 다른 코드를 새로 작성해야 하는 불편함이 있다.

가장 간단한 해결책은 키음 슬롯의 자리 수를 두 자리 이상으로 확장하는 것이다. 변환 과정도 간단하며, 기존의 BMS 처리 코드를 약간만 변경하면 적용할 수 있는 강점이 있다. 키음 슬롯 자릿수를 BMS 파일 내부에서 #SLOTLENGTH 3 같은 문구로 기술하고 구동기 측에서 (기본값을 2로) 위 문구를 읽고 처리하면 될 것이다. 사실 키음 슬롯이 세자리만 되어도 001-FFF까지 46655개이며 이는 충분하고도 남는 숫자이다. 이 이상을 활용하고자 하면 BMS 데이터를 읽는 데에만 곡 길이보다 더 긴 시간이 걸릴 것이다.

**2. 작성 가능한 마디 개수의 제한**

마디 번호에 할당된 자릿수가 10진법으로 3자리이기 때문에 #999 이후의 마디를 기술할 수 없다. BPM과 마디 길이의 기본값인 120과 4/4를 적용하면 2000초로 음악의 길이가 제한된다. (일반적인 BMS는 대략 2분 전후의 길이를 가진다.) 2000초면 충분하게 생각될 수 있으나, BPM이 2000인 스피드코어를 BMS로 작성하려 하면 길이가 120초로 제한되는 셈이다. 물론 마디의 길이를 자유롭게 늘릴 수 있고, BPM을 자유롭게 설정할 수 있기 때문에 비상식적으로 빠르거나 길이가 긴 BMS 파일을 작성하려면 못할 것도 없다. 이는 사용할 수 있는 천공 카드의 개수는 제한되어 있으나, 천공 카드의 길이와 읽어가는 속도를 자유롭게 할 수 있음에 비유할 수 있다. 실제로 이런 특징을 최대한 활용하여 1주일동안 쉬지 않고 목탁을 치는 체험을 할 수 있는 MOKUGYO ALLWEEK이라는 실험적 작품도 나왔다. (이후 후속작으로 6개월간 목탁을 두드리는 MOKUGYO AllCenturyMIX도 나왔다. 당연히 양측 모두 완주자는 커녕 5시간 버틴 사람이 최고기록으로 남아있다.)

문제는 BMS를 이용해 시각적 효과를 삽입하려 할 때 발생한다. 마디와 마디 사이의 경계선인 마디선은 일반적인 경우 하얀 수평선으로 노트와 함께 낙하하는데, 이 수평선을 활용한 연출을 할 경우 마디의 수가 부족할 수 있다. 예를 들면 마디선이 위로 올라가는 연출을 넣고 싶다고 하자. 일반적인 경우 마디선은 항상 아래로 내려오고, BMS는 역방향으로 읽을 수 없으므로 불가능한 것처럼 보인다. 하지만 초당 24프레임으로 각각 다른 마디선의 위치를 나중에 있을수록 위로 올라가게 설정하면 (각각의 마디선 근처에서 낙하를 1/24초간 정지시키고, 마디선 사이에서는 내려오는 속도를 매우 빠르게 설정한다.) 잔상 효과에 의해 올라가는 것처럼 보이게 할 수 있다. 문제점은 이 효과를 5초 정도만 넣고자 해도 마디선이 120개가 (즉 120개의 마디가) 필요하며, 복수의 마디선으로 조금 더 화려한 연출 (마디선을 교차하는 것처럼 보이게 등)을 시도하면 마디의 개수가 필히 부족해진다.

마디 번호에 36진법을 도입하려는 시도를 할 수 있으나, 이는 현재까지의 BMS와 호환이 되지 않으며 (#000을 고려하지 않을 시 #010은 현재 10번째 마디이나, 36진법 적용시 36번째 마디가 된다.) 타 명령어(#WAVXX 등)와 겹치는 위험이 있다. 시간 계산이 불편한 문제도 있다. 한 마디의 길이가 2초일 때 마디 번호 #00P에 도달하는데 걸리는 시간을 직관적으로 계산할 수 있겠는가?

의외로 해결 방법이 간단한데 #999XY 다음 마디를 #1000XY 으로 작성하는 것이다. 구동기가 마디 번호 영역을 # 이후의 세 글자가 아닌 #과 : 사이의 마지막 두 글자를 제외한 것으로 처리하게 만들면 되기에 구동기에 적용하는 과정도 어렵지 않다. 이럴 경우 마디선을 이용한 연출이 더 자유롭고, BPM이 2000 이상인 실험적인 장르의 BMS 도입을 기대할 수 있다.

**3. Lunatic Rave 2: BMS와 MD5**

Lunatic Rave 2의 강점 중 하나를 인터넷 랭킹 지원으로 소개한 바 있다. 그 수많은 BMS 파일을 서버 측은 어떻게 구분하는가? 플레이에 사용된 BMS 파일을 통째로 업로드하는 것이 아닌, BMS 파일을 입력으로 받아 128비트의 해시를 출력하는 MD5를 사용하여 그 해시로 구분한다. 아직까지 유의미한 실행 결과를 보이는 서로 다른 BMS 파일의 해시 값이 같은 경우는 보고되지 않았으므로 이는 효율적인 방법으로 보인다.

다만 이 방법은 Lunatic Rave 2의 또다른 강점인 BMS 난이도표(약 1700개의 특정 BMS 파일의 난이도를 BMS 유저 간 투표를 거쳐 정리한 표이다)의 기준 구동기라는 점과 마찰을 겪었다. 2012년 키음 배열이 원곡과 다른 (원곡 작성자가 작성하지 않은) 패턴들을 난이도표에서 삭제 후, 원곡과 같도록 수정한 패턴을 난이도표에 등재하였다. 키음이나 배경음 배열의 사소한 차이 이외에는 두 패턴은 동일하지만, 인터넷 랭킹은 MD5를 사용하는 관계로 두 패턴의 플레이 기록이 서로 호환되지 않았다. 이에 대한 비판이 제기되었고, 최근에도 가끔씩 몇몇 패턴이 수정 후 등록되는 일이 발생하지만 인터넷 랭킹 측에서 별도의 조치를 취하지는 않았다.

음악 파일의 배열과 관계없이 연주해야 하는 노트 배열과 내려오는 속도가 동일하면 동일한 패턴으로 취급하는 것이 상식적이다. 이 문제를 해결하기 위해서는 BMS 파일 자체가 아닌 다른 부분을 MD5에 입력해야 할 것이다. 키음 영역의 노트에 대해서 노트를 입력해야 하는 키와 시각, BPM 변화의 리스트를 작성 후 이 리스트를 MD5에 입력하여 얻은 해시값을 인터넷 랭킹에 사용한다면 이 문제는 해결될 수 있다. 이는 과거에 제작된 BMS에 그래픽 요소를 (배경에 출력되는 그림이나 영상) 추가하여 재배포한 경우 두 파일이 인터넷 랭킹에서 호환되는 장점도 존재한다. 사실 가장 간단한 방법은 인터넷 랭킹 측에서 수정 전후의 파일의 기록이 호환되도록 지원하는 것이나 이는 요원해 보인다.

**IV. 결론**

BMS의 변천사를 아케이드 리듬게임의 역사와 결부하여 설명하였고, BMS의 표준화 실패 사례를롱노트의 도입과 BMS에서의 기술과 처리 방법을 예시로 논의하였다. 또한 리듬게임의 발전이 멈춰도 BMS 포맷과 구동기의 발전이 필요함을 그 방향과 함께 역설하였다.

실질적 표준인 Lunatic Rave 2가 신기능을 지원한다면 BMS의 제작 자유도 또한 그만큼 증가하게 된다. 그러나 Lunatic Rave 2의 업데이트가 2010년 이후로 전무하고, 이것이 10년 가까이 표준으로 자리잡은 것이 문제의 경직성에 크게 기인하고 있다. 제작자 측은 프로그램 코드를 분실하였다는 해명을 하였지만, 9년동안 아무런 결과를 내놓지 않는 것은 분명 무책임한 모습이다. BMS는 그 구조가 공개되는 포맷이나 Lunatic Rave 2를 포함한 대다수의 구동기는 오픈소스가 아니라는 점이 발전의 걸림돌이 된다. 이러한 경직성에서 탈피하기 위해 Lunatic Rave 2를 대체할 새로운 BMS 구동기를 제로부터 개발하거나 오픈소스 프로젝트를 진행하고 있다. 하지만 Lunatic Rave 2의 시스템을 그대로 옮겨온 상위호환이 개발되지 않는 이상 앞으로도 표준이 변할 일은 없어 보임이 아쉬울 뿐이다.

**참고 사이트**

“BMS command memo (draft)”, 2014. Retrieved from https://hitkey.nekokan.dyndns.info/cmds.htm

“HOW TO PLAY beatmania”, 1998. Retrieved from <http://www.konami.jp/am/bm/1st/how/1sthow.html>

“遊び方について, beatmania IIDX 26 Rootage”, 2018. Retrieved from  
https://p.eagate.573.jp/game/2dx/26/howto/play/game\_start.html