한국음향학회지 제24권 제7호 pp. 365~370 (2005)

Csound를 이용한 음악 프로그래밍 언어 제작

Music Programming Language Composition Using Csound

여 영 환*
(Young-Hwan Yeo*)

*숙명여자대학교 음악대학 작곡과

(접수일자: 2005년 3월 31일; 수정일자:2005년 6월 14일; 채택일자: 2005년 8월 30일)

본 논문은 Csound 프로그래밍 언어에 대한 발전된 학술 연구 혹은 독창적인 학술 아이디어를 탐구하기에 앞서, 현재 한국에서는 학술적 연구나 소개가 전혀 이루어지지 않은 Csound 음악 프로그래밍 언어를 누구도 쉽게 접근할 수 있도록 체계화된 이론을 정립하고 보급하기 위해 제안한 것이다. Csound는 세계적인 컴퓨터 음악음향 제작 프로그래밍 언어이며, 세계적으로 유명한 사운드 디자이너들이 이용하는 전문적인 텍스트 기반 소프트웨어 신디사이저로, 1985년 M.I.T. Media Lab의 Barry Vercoe에 의해 개발되었다. 본 본문은 서양 전통음악의 역사적인 관점에서 음악과 자연의 소리 혹은 특정 매체로부터 발생되는 소리와의 결합시기를 전자음악과 음악음향 제작의 출발점으로 제시하였다. 그리고 기초적인 Csound의 작동원리를 서술하고, 이를 기초로 Csound를 이용한 음악 프로그래밍의 적용예를 제시하였으며, 결론에서는 본 논문의 연구 목표와 앞으로의 연구 과제를 서술하였다.

핵심용어: 씨사운드, 음악프로그래밍 언어, 음악음향, 컴퓨터 소리합성, 컴퓨터음악, 전자음악, 텍스트 기반 신디사이저

투고분야: 음악음향 및 음향심리 분야 (8.1)

The present study is purposed to establish a systematic theory for user-friendly approach to the creation of using a programming language using Csound. Csound is a world-wide computer music programming language and a software synthesizer specialized for prominent sound designers developed by Barry Vercoe at the Media Laboratory in M.I.T. The introduction and the main body of this paper suggested as the starting point of creating electronic music and musical sound the time of combination of music with natural sound or sound from specific media from the viewpoint of traditional Western music, and presents a systematic method composed of the principle of the operation of Csound and basic data samples.

Keywords: Csound, Music programming language, Musical Acoustics, Computer sound synthesis, Computer music, Electronic music, Text based synthesizer.

ASK subject classification: Musical Acoustics and Psychoacoustics (8.1)

I. 서 론

컴퓨터 프로그래밍 언어로서 Csound를 이용한 음악음향 제작과 컴퓨터 소리합성 (computer sound synthesis)을 논하기에 앞서 모든 소리에 대한 연구는 자연의 소리로부터 출발되어야 된다. 사람의 귀에 들리는 모든 소리는 진동체 혹은 특정 매체가 진동 (vibration)하여 물과 공기와 같은 매질에 압력변화를 주며, 상온 0도 일 경우 음속은 약 333.4m이며, 온도가 1도 상승할 때마다 음속은 대략 0.6m씩 빨라지게 된다. 이러한 특정 매체 등에서 생산된 소리는 공기와 같은 매질을 통해 공기밀도의 변화를 전달하여 귀의 고막을 흔들게 되고, 우리는 그것을 소리로 지각하게 된다.[1,7,8]

이러한 자연의 소리 혹은 특정 매체로부터 발생되는 소리들을 음악의 재료 혹은 음악음향으로 이용한 최초의 시점은 20세기 초 이탈리아를 중심으로 일어난 미래주의 (未來主義, Futurism) 작곡가부터라 할 수 있다. 이것은 전통적인 형식과 미 (美)를 거부하고, 새로운 세대에 맞는 소재와 테마의 확립을 이념으로 흥미 있는 색채감이 두드러지는 것이 특징이며, 루솔로 (Luigi Russolo, 1885-1947)의 소음음악이 대표적이라 할 수 있다. 루솔

책임저자 여 영 환 (younghwanyeo@sookmyung.ac.kr)
140-742 서울특별시 용산구 청파동 2가 53-12 숙명여자대학교 음악대학 작곡과(309호)
(전화: 02-710-9584; 팩스: 02-710-9584)

로는 1913년 선언문인 "소음예술 (The Art of Noises)" 를 발표하였다. 이 선언문에서 루솔로는 일상생활에서 녹음할 수 있는 다양한 효과음과 소음, 물리적인 효과로 존재하는 기존 소리를 모방하는 것, 기계 작동에서 나오는 다양한 소음, 기차가 달릴 때 들려지는 다양한 소리, 그리고 전쟁에서 채집할 수 있는 새로운 소음 및 음향 등을 음악과 결합하여 사용하는 것을 목표로 하였다. 또한 1921년 파리에서 개최된 그의 음악 콘서트 (music concert)에서는 미래주의 음악이 어떤 것인지 단적으로 청중들에게 보여주며, 음악과 소음의 결합을 통한 새로운 미래 음악 음향 예술의 가능성을 보여 주고자 하였다.[7]

이와 같이 미래주의 작곡가들이 최초로 시도하였던 소음음악은 1948년 프랑스 방송국의 엔지니어였던 피에르 쉐퍼 (Pierre Schaeffer, 1910-1995)에게 결정적인 영향을 주게 되었고, 그에 의해 구체음악 (具體音樂, Musique Concrete)이라는 새로운 장르가 시도되어 독일의 전자음악 (Electronic Music)과 미국 중심의 컴퓨터 음악 (Computer Music)의 발전에 시발점을 제시하였다.

1950년경 미국을 중심으로 출발한 컴퓨터음악은 프랑스의 구체음악과 독일의 전자음악과는 달리 컴퓨터를 이용해 새로운 음색 (timbre)을 창출 하고자 하였다. 비록 컴퓨터는 약 50-60년 정도의 짧은 역사에 비해 시스템 (system)은 현재까지 급속히 발전 보완되고 있다. 이러한 컴퓨터의 급격한 발전은 컴퓨터 음악음향 제작가들에게 많은 도움을 주었으며, 이와 함께 성장하고 개발된 소프트웨어 (software) 및 하드웨어 (hardware)들의 비중이 음악 음향 제작기술에 많은 부분을 차지하게 되었다.[4]

컴퓨터를 이용한 음악음향 제작 도구의 본격적인 연구 개발 시점은 1960년대 맥스 매듀 (Max Mathew)에 의해 개발된 Music 4가 스탠포드 대학교와 프린스턴 대학교에서 본격적으로 연구되면서부터 출발하였으며, 많은 연구 개발과 시행착오 과정을 거쳐 1985년 M.I.T. (Massachusetts Institute of Technology) Media Lab의 Barry Vercoe에 의해 최종적으로 Csound가 완성되었다.[1-2] Csound는 기존의 신디사이저와는 달리 컴퓨터 텍스트 (text) 명령을 바탕으로 한 소프트웨어 신디사이저 (software synthesizer)로 다양한 음악음향 및 전자음향 제작이 가능하여 많은 음악 음향 제작가들에게 필수적인 제작 도구로 이용되고 있으며, 리눅스와 같은 공개 프로그램이므로 현재까지 지속적으로 많은 프로그래머들과 M.I.T. 연구진들에 의해 추가 기능이 개발되고 있으므로 앞으로 더욱 강력한 기능한 가진 텍스트 기반 소프트웨어 신디사이저로 발전될 것으로 예상된다.[4-6]

따라서 본 논문은 기본적인 Csound의 작동원리를 제시하고, 이를 기초로 Csound를 이용한 음악 프로그래밍의 적용 예를 제시하여 앞으로의 연구과제를 제시하는 것이 연구목적이다.

II. Csound의 작동원리

2.1. Csound 개요

Csound는 1985년 M.I.T. (Massachusetts Institute of Technology) 교수인 Barry Vercoe에 의해 개발된 디지털 사운드 합성 프로그램 (digital sound synthesis program)으로서 원하는 모든 디지털 형태의 소리와 음악음향 효과들을 다양한 운영체계 속에서 모노 (mono, 1채널) 혹은 스테레오 (stereo, 2채널) 등 원하는 사운드 채널로 생성할 수 있는 텍스트 명령 기반의 소프트웨어 신디사이저 프로그램이다.[1-3]

Csound의 사용환경은 MAC (macintosh) 혹은 PC (personal computer)에서 모두 가능하며, 컴퓨터 사양만 뒷받침 된다면 음악음향 및 전자음향 생성을 위한 데이터 처리 속도는 무척 빠르다. 이 프로그램은 컴퓨터 C 언어를 기반으로 제작되었으며, ANSI-C언어를 기본으로 개발되어 각 컴퓨터 오퍼레이팅 시스템에서 컴파일 (compile)만 하면 사용될 수 있다. 또한 Csound의 활용을 위한 C 언어 선행학습은 불필요하며, Csound의 작동원리를 이해하고 ORC (orchestra, 오케스트라) 파일과 SCO (score, 스코어) 파일 작성에 대한 충분한 이론학습과 경험이 뒷받침 된다면 어느 누구도 훌륭한 음악음향을 제작할 수 있다.[4]

많은 기간을 거쳐 보완 수정된 Csound는 약 400개 이상의 신호처리 모듈 (module)을 이용하여 신디사이저를 만들 수도 있고, 멀티 이펙트 프로세서 (multi effect processor)로도 이용할 수 있다. 이는 사용자가 원한다면 생성하고자 하는, 즉 머릿속에서 상상하는 모든 디지털 형태의 음악음향들을 제작할 수 있음을 말한다.

2.2. Csound의 작동원리

기본적으로 컴퓨터는 복잡한 형식의 이진수 (binary numbers)들로 구성된 "기계적인 언어 (machine language)"만을 이해할 수 있다. 하지만 Csound이용자들은 선행

학습을 위해 Basic, Pascal, Fortran등과 같은 높은 수준의 컴퓨터 언어를 학습 할 필요는 없다. 이는 기존 컴퓨터 언어와 기본적인 개념과 목적으로 비추어 볼 때 Csound는 전자음향의 창출 그리고 합성을 위해 개발된 특수한 컴퓨터 프로그래밍 언어 분야라고 생각하면 될 것이다.

일반적으로 컴퓨터를 이용한 디지털 음악합성은 상당히 복잡하며, 각 초 마다 생산하려는 소리의 산출하기 위해 수천 번의 기계적인 계산이 요구되며, 이의 실행을 성공적으로 수행하기 위해 사용자에게 정확한 명령과 지시를 요구한다. 다행히 프로그래머들에게 수많은 기계적 산출과 계산을 위해 개별적이며 동일한 명령의 반복을 피할 수 있는 다양한 방법들이 있다. 예를 들어 프로그램 실행에서 한번 이상의 관련된 반복된 명령은 "서브루틴 (Subroutine)" 이라 불리는 기능에 의해 자동적으로 동일 명령을 반복 실행할 수 있다. Csound 언어에서 볼 수 있는 많은 진술 (statements)들을 구성하는 특정 명령어는 생산하려는 소리의 산출하기 위해 수천 번의 기계적인 계산 혹은 명령 반복을 서브루틴을 통해 해결한다.

앞에서 밝혔듯이 Csound는 다양한 운영체계에서 운영되지만 본 논문에서는 PC console version 중심으로 설명하고자 한다. 먼저 Csound 프로그램을 실행하기 위해 사용자들은 텍스트 에딧트 프로그램을 이용하여 두 개의 파일(file)들을 작성해야 된다. 이는 '오케스트라 (orchestra)' 와 '스코어 (score)다. 스코어는 사용자가 음의 음정(pitch of note), 음의 길이 (tone duration), 음의 시작시간 (starting time), 그리고 저장된 웨이브 모양 (stored waveshapes)과 같은 데이터를 결정할 수 있는 실질적인 부분이다.[3]

오케스트라 파트 역시 사용자들이 자신이 원하는 명령을 실행시키기 위해 구성되는 실질적인 파일로써 스코어를 연주하기 위한 악기부분이라 말할 수 있다. 오케스트라는 헤더 (header) 부분과 바디 (body)로 구성되며 스코어 파일의 데이터를 바탕으로 연주될 때 컴퓨터에게 특정 관련 임무를 지시하기도 한다.

에딧패드 라이트 (EditPad Lite)와 노트패드 (NotePad) 등과 같은 텍스트 편집 (text edit) 소프트웨어를 이용하여 오케스트라 파일과 스코어 파일을 작성하고

1) The orchestra (file extension: .orc)
2) The score (file extension: .sco)

그림 1. Csound의 파일의 구성 - 오케스트라/스코어
Fig. 1. Formation of Csound file - orchestra/score.

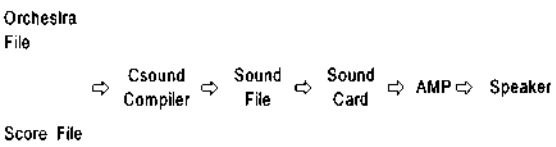

그림 2. Csound의 원리
Fig. 2. Principle of Csound.

Csounder[1]에서 각 파일을 인식할 수 있도록 약속된 파일 확장자 (file extension)를 이용하여 저장한다. 완성된 각 파일들은 Csounder를 통해 wave나 aiff 형태 등의 원하는 오디오 파일 (audio file) 형태로 생성 될 수 있으며, 웨이브 편집 소프트웨어와 연동하여 생성된 파일을 편집할 수도 있다.

Csound에 사용되는 대부분의 진술들의 목적은 컴퓨터가 어떠한 명령을 실행될 수 있도록 각 명령에 대한 구성요소를 서술하는 것이다.[1]

일반적으로 Csound를 통해 사운드를 창출하기 위해서 컴퓨터는 사용자에게 하나 혹은 그 이상의 옵코드 (opcode), 매개변수 (parameter) 그리고 이에 따른 아규먼트 (argument)를 요구한다.

Csound에 명시된 인풋 (input)과 아웃풋 (output)들은 상징적인 형식 (symbolic form), 즉 사용자에 의해 주어진 이름에 의해 자유스럽게 명기 될 수 있으며, 구성요소인 진술(statement)은 통상적으로 다음과 같은 3개의 요소들에 의해 구성된다. - 실행되어야 될 작업의 이름 (the name of operation), 인풋으로 사용될 값 (value), 아웃풋 혹은 결과 (result)의 값.

결과를 산출하기 위해 Csound에서 사용되는 진술들을 논리적으로 서술하는 방법은 대부분의 컴퓨터 프로그래밍 언어들과 같이 오른쪽에서 왼쪽으로 가술되어야 된다.[3]

① 오케스트라 (Orchestra)의 형식

Csound의 오케스트라 파트는 컴퓨터에게 명령 지시를

Output (Result) ← Operation ← Input

그림 3. 진술(statement)의 형식
Fig. 3. Formation of statement.

ORCHESTRA

HEADER

BODY: INSTRUMENT(S)

그림 4. 오케스트라 파트의 구성
Fig. 4. Formation of orchestra part.

1) Compiler, 오케스트라 파일과 스코어 파일을 컴파일(compile)하기 위한 프로그램

하는 진술들이 논리적으로 정리된 행이라고 할 수 있다. 이는 크게 헤더 (header)부분과 악기들로 구성되는 바디 (body)부분으로 크게 나눌 수 있다.

ⓐ HEADER

오케스트라 헤더 부분은 모든 악기부분 (instruments part)과 공유하는 4개의 기본적인 변수 (variables)로 구성되며 반드시 오케스트라 부분의 첫 시작 부분에 기술되어야 된다. 헤더 부분은 다음과 같은 진술로 구성되며, 통상적으로 4부분의 매개변수들로 이루어져 있다.

ⓑ BODY: INSTRUMENT(S)

헤더부분은 4개의 약속된 언어만을 사용하지만, 악기 부분은 보다 복잡하다. 복잡한 정도는 충족하고자 하는 프로세스에 따라 다르다. BODY 부분은 오케스트라를 구성하는 하나 혹은 그 이상의 "가상악기 (virtual instrument)" 로 구성되며, 각 악기 부분들은 스코어 (score)부분의 데이터와 컴파일하여 최종 사운드를 창출하기 위한 많은 진술로 구성되어 있다.

위의 예시에서 만들어지는 사운드는 amplitude 10000, freq. 220hz (가온 A보다 한 옥타브 낮은 소리), 파형번호 1을 충족시키는 소리며, 결과물을 나타내는 asig의 첫 자인 "a"는 asig가 "audio" variable임을 의미한다.[1-3]

ⓒ 진술 (statement)의 구성

오케스트라의 진술은 다음 주어진 형식과 같이 최소 5개 이상의 분야 (field)로 구성된다.

오케스트라 파일의 작성에서 라벨, 결과, 옵코드 등의 각 분야는 최소한 한 스페이스 (space) 이상의 공간으로만 구분되면 아무런 에러 (error)가 발생하지 않지만, 탭키 (tab key)를 사용하여 각 항목의 구분을 추천한다. 왜냐하면 종적으로 또는 횡적으로 각 행의 진술 분야 (statement field)를 시각적으로 잘 배정하는 것은 추후에 사용자가 오케스트라 파일의 전체적인 모습을 보기 용이하며, 에러 수정에 편리하기 때문이다. 참고로 커멘트 (comment)란은 사용자가 그 행을 명령수행을 요약할 수 있는 공간으로 추후에 자신이 작성한 그 행의 리뷰 (review)하기에 용이하도록 설명을 할 수 있는 공간이다. 커멘트 란은 반드시 세미콜론으로 시작되어야 되며, 세미콜론 뒤의 어떠한 명령 혹은 문장도 컴퓨터는 인식하지 못한다.

㉠ Statement Syntax의 구성

✔ Label field: 진술 라벨 (statement label)은 사용자의 선택 옵션이다.

✔ Comments field: 커멘트 란은 반드시 세미콜론으로 시작되어야 되며, 세미콜론 이하의 어떠한 명령 혹은 문장은 컴퓨터가 인식하지 못한다.

㉡ Basic Statement의 구성

✔ Result field: 결과란은 기본 진술 (basic statement)에서 산출되는 결과를 의미하는 베리어블 (variable) 이름으로 작성된다. 베리어블의 이름은 숫자와 영문자의 조합으로 이루어진 알파뉴메릭 (alpha- numeric) 형태로 만들어야 된다.

㉢ Operation and arguments의 구성: 오퍼레이션 부

Formation :

✓ sr: sampling rate
✓ kr: control rate
✓ ksmps: sr/kr ratio
✓ nchnls: number of output channels

Example :

✓ sr=44100
✓ kr=4410
✓ ksmps=10
✓ nchnls=1 (혹은 2)

그림 5. 일반적인 헤더 파트의 구성과 예
Fig. 5. General formation and example of header part.

Formation :

instr

(Body of the instrument)

endin

Example :

```
          instr     1
asig      oscil     10000,220,1   ;크기,주파수,파형번호
          out       asig          ;소리의 출력(모노)
          endin
```

그림 6. 오케스트라 바디의 구성과 예
Fig. 6. Formation and example of orchestra body.

Label: result opcode argument1,argument2,...; comments

그림 7. 오케스트라 진술의 구성
Fig. 7. Formation of orchestra statement.

분은 반드시 Csound의 컴멘트 혹은 "오퍼레이션" 명령어를 포함해야 된다. 아규먼트 (arguments) 부분들은 컴멘트 혹은 아규먼트에 관련된 실행 정보 (information)를 포함해야 된다. 아규먼트는 컴머(,)에 의해 구분되며 구성하는 명령어 사이는 컴머를 사용하는데 각 아규먼트 사이에는 빈 공간(blank space)은 통상적으로 넣지 않는다.

② 스코어 (Score)의 형식

오케스트라 파일처럼 스코어 파일도 통상적으로 Functions와 Notes 두 부분으로 나뉘어 진다.

ⓐ 기능 진술 (function statements): 이는 사인 웨이브와 같이 특정한 웨이브 폼 (wave form)을 이용하기 위해 사용된다. 만약에 샘플된 소리들을 이용할 경우에는 스코어 파일에서 이 기능을 사용하지 않을 수도 있다.

다음 예시에 나오는 f 진술 (statement)은 사인 웨이브 폼을 창출한다.

㉠ Statement 의 구성

✔ f1: Function ID 번호를 말한다.

✔ 0: 이 기능을 실행하는 실행시간을 말한다. 만약 0이 아닌 3이라면 이 기능은 스코어 파일이 실행된 3초 후에 실행된다.

✔ 4096: 이 웨이브폼 안에 있는 테이블 사이즈 (table size)를 말한다. 다시 말하면, 웨이브폼은 4096개의 기억장소 (memory location)에 배열된다. 기억장소는 "2의 거듭제곱 +1" 로 계산되며, 반드시 2의 20 거듭제곱을 초과해서는 안 된다. (257, 513, 1025, 4097 ..... 16777216)

✔ 10: f functions는 웨이브 폼을 생산할 수 있는 많은 방법 중 하나를 이용해야 된다. 대표적인 방법은 젠 루틴 (GEN routines)을 이용하는 것이며, 각각의 젠 루틴 (GEN routines)은 고유의 ID 번호를 가지고 있다.

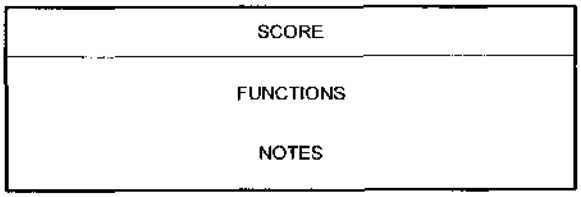

그림 8. 스코어 파트의 구성
Fig. 8. Formation of score part.

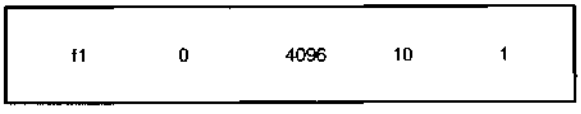

그림 9. f statement의 예
Fig. 9. Example of f statement.

위 fig. 8의 예시에서는 3개의 사인파의 하모닉스 비율을 기대할 수 있다.

(fundamental, second harmonic, third harmonic), all amp level=1.

ⓑ 노트 (악보, notes): 스코어 파일에서의 노트의 역할은 절대적으로 필요한 존재이다. 통상적으로 피아노와 같은 실제 악기를 위한 사보음악에 나오는 악보와 같이 Csound에서는 생각하지 않는 것이 도움이 될 것이다. 왜냐하면 실제 음악에 쓰이는 악보와 달리 모든 스코어의 정보는 숫자로 음정, 음의 길이, 음의 시작 시간 등 모든 것을 결정하기 때문이다. 노트들은 악기진술 (instrument statements, i statements)에 의해 생산된다. i 진술은 오케스트라 파일에 있는 제시한 숫자들과 결합하여 악기들을 실행시킨다.

III. Csound를 이용한 음악 프로그래밍 언어의 적용

다음은 Csound 프로그래밍 언어를 이용하여 아래의 1~6 항목을 충족하는 컴퓨터 음향을 제작한 예이다.

1. 샘플링 레잇 (sampling rate)[4] 44.1khz - CD format
2. stereo type - 2채널[5]

```
<Example.orc>
sr      =       44100       ;sampling rate, 44100hz
kr      =       4410        ;control rate[1]
ksmps   =       10          ;sr/kr[2]
nchnls  =       2           ;stereo
        instr1              ;beginning instr.
asig    oscil   7500,440,1  ;play function1 at A-
        outs    asig,asig   ;stereo out
        endin               ;end
```

Fig. 10. ORC 파일
Fig. 10. ORC file.

2) 아날로그 신호가 디지털로 전환될 때 초당 얻어지는 샘플의 수를 말한다. 샘플링 레이트가 높을수록 표현 가능한 주파수의 대역이 높아진다. 예를 들어, 44100hz는 0.00002초당 샘플 1개씩 읽혀지게 된다.

3) 1=mono, 2=stereo, 4=quadraphonic

```
<Example.sco>

;generate function number 1
f1 0      512     10      1          ;sine wave in 512-locations
;play one note starting at time 0 and lasting 1 beat
i1 0      5
e                                    ;end score
```

Fig. 11. SCO 파일
Fig. 11. SCO file.

3. amp 7500[6]
4. 연주 음정 (tone pitch) – 가온 A (440hz),
5. 음색(tone color) – sine wave (pure tone)
6. 음의 길이(tone duration) – 5초

IV. 결 론

1950년경부터 서양전통음악의 연장선으로 발전된 음악음향 분야는 과학기술의 발달로 소프트웨어 및 하드웨어적 음원모듈 개발과 함께 많은 발전을 이루고 있으나, 사실상 사용자들은 회사 소속의 사운드 디자이너에 의해 개발된 제한된 사운드 음원들에만 만족해야 되어 사실상 자신들이 원하는 색채의 음악음향만을 사용할 수 없는 실정이다.

본 논문은 Csound 프로그래밍 언어에 대한 발전된 학술 연구 및 독창적인 학술 아이디어를 보여주기에 앞서, 현재 한국에서는 기초적인 학술 연구뿐만 아니라 한글 매뉴얼조차 없는 Csound 음악 프로그래밍 언어를 초보 음악음향 제작자들도 쉽게 접근할 수 있도록 체계화된 이론정립과 보급 확대를 위해 제안 되었다.

이 논문을 계기로 지속될 추후 연구 방향은 Csound 프로그래밍 언어에 대한 많은 이론들을 한국에 체계적으로 보급하고, 다양한 음악음향 제작법을 제시하여 Csound를 이용하고자 하는 음악음향 제작자들에게 실질적인 가이드라인을 제시하고자 한다.

참 고 문 헌

1. R. Boulanger, ed., *The Csound Book*, (The MIT Press, Cambridge, Massachusetts, 2000).
2. R. Boulanger, *Making Music With Csound*, (The MIT Press, Cambridge, Massachusetts, 1990).
3. R.Bian chini and A.cipriani *Virtual Sound* (contemp s.a.s,Rome, 2000)
4. C. Roads, *The Computer Music Tutorial*, (The MIT Press, Cambridge, Massachusetts, 2000).
5. P. Griffiths, "A guide to Electronic Music," The Pitman Press, Bath, 1979.
6. S. Pellman, *An Introduction to the Creation of Electroacoustic Music*, (Wadsworth Publishing Company, Belmont, California, 1994).
7. J. Chadabe, *Electric Sound*, (Prentice Hall Inc, Upper Saddle River, New Jersey, 1997).
8. C. Dodge, and T. A. Jerse, *Computer Music*, (Schirmer Books, New York, 1985).

4) CD format 16bit의 최대 음량(amp): 32767

저자 약력

• 여 영 환 (Young-Hwan Yeo)

2004.03~현재: 숙명여자대학교 음악대학 작곡과 조교수
(주) 투윈미디어-문화콘텐츠진흥원 녹음실 기술이사
2003.03~2004.02: 천안대학교 음악학부 전임강사
2003.01~05: 포항공과대학교 컴퓨터공학과 가상현실 연구소 방문연구원
2003.12.20: University of Texas at Austin, 전자음악/컴퓨터음악 박사학위 취득 (D.M., Emphasis in Electronic/Computer Music)
2003.12.20: University of Texas at Austin, 작곡전공 박사학위 취득 (D.M.A. Composition)
2001.05.19: University of Texas at Austin, 작곡전공 석사학위 취득(M.M. Composition)
1999.02: 경북대학교 예술대학 음악학과, 작곡전공 학사학위 (B.A. Music) 취득
※주관심분야: 전자음악/컴퓨터음악 (Electronic Music/Computer Music) 작곡, 음악음향/전자음향 제작, 음악 프로그래밍 언어 (Csound), 컴퓨터 소리합성, 디지털 오디오 워크스테이션 (DAW, Digital Audio Workstation), 사운드 라이브러리(Library), 영상음악 / 사운드 이펙트 (Effect)