박 사 학 위 논 문

웹 기반의 오디오 라이브러리 개발을 통한 인터랙티브 웹 애플리케이션 제작 연구

지도교수 김 준

동국대학교 영상대학원 멀티미디어학과

홍 의 식

2 0 1 9

박 사 학 위 논 문

웹 기반의 오디오 라이브러리 개발을 통한 인터랙티브 웹 애플리케이션 제작 연구

홍 의 식

지도교수 김 준

이 논문을 박사학위논문으로 제출함

2019년 6월

홍의식의 음악박사(컴퓨터음악전공)학위 논문을 인준함

2019년 7월

위원장 박 상 훈 (인)

위 원 여 운 승 (인)

위 원 이 교 구 (인)

위 원 최 홍 찬 (인)

위 원 김 준 (인)

동국대학교 영상대학원

목 차

표 목 차

그 림 목 차

Ⅰ. 서론

1. 연구 배경 및 목적

페이스북(Facebook)[1] 최고경영자인 마크 저커버그(Mark Zuckerberg)는 2017년 6월, 본인의 페이스북 계정을 통해 페이스북 월 사용자가 20억 명을 넘었다고 발표하였다. 출시 13년 만에 10억 명을 넘었고, 5년 뒤 20억 명을 돌파하였는데 이는 전 세계 인구를 약 75억 명으로 감안했을 때 놀랄만한 수치이다 [1].

페이스북은 사용자들 간의 정보 공유와 커뮤니케이션(communication) 형성을 위한 대표적인 소셜 네트워크 서비스(SNS, Social Network Service) [2]로 시간과 장소에 구애받지 않고 서비스를 유지할 수 있는 온라인 플랫폼(online platform) 환경을 제공한다. 이와 같은 소셜 네트워크 서비스는 월드 와이드 웹(World Wide Web)[2] [3]의 발전과 모바일의 확산, 무선 네트워크 속도 등에 큰 영향을 받으며 급속도로 성장하였다. 특히 웹에서 제공하는 기술은 소셜 네트워크 애플리케이션(application)의 성능을 증대시켰고 오디오와 이미지, 비디오 등의 다양한 멀티미디어 플랫폼 구현을 가능하게 하였다.

이러한 웹은 20여 년 전 문서 공유를 위해 개발되어 현재 4차 산업시

1) 2004년, 하버드 재학생이던 마크 저커버그가 창립한 소셜 네트워크 서비스.
2) 인터넷을 통해 정보를 공유할 수 있는 공간.

대의 핵심 기술을 주도하고 있다 [3]. 4차 산업의 주요 기술은 빅 데이터(Big Data) [4], 사물인터넷(Internet of Things) [5], 인공지능(Artificial Intelligence) [6] 등이 제시되고 있으며, 웹 기반의 플랫폼과 융합하여 발전하고 있다 [7]. 특히 웹은 접근성이 뛰어나 모바일과 PC를 통해 다양한 정보와 응용 소프트웨어 플랫폼, 문화, 예술 등의 사회 전반에 걸친 새로운 융합콘텐츠를 제공한다. 하지만 불과 몇 년 전만 해도 웹은 다음과 같은 문제를 제시하며 새로운 해결방안을 필요로 하였다.

- 크로스 브라우징(cross browsing)[3] 호환성
- 멀티미디어 콘텐츠 사용을 위한 플러그인 설치
- 웹 브라우저 실행 속도

2008년 W3C(World Wide Web Consortium)[4]은 크로스 브라우징 호환의 이슈를 해결하기 위해 웹 표준을 제안하였으며, HTML5를 정식 웹 표준으로 채택하였다 [8]. 이러한 HTML5는 강력한 API(Application Programming Interface)를 제공하여 추가 플러그인 없는 웹 기반의 멀티미디어 콘텐츠(multimedia contents) 애플리케이션 개발을 가능하게 하였다 [9]. 같은 해 구글(Google)은 크롬(Chrome) V8 엔진 [10]을 발표하면서 자바스크립트 [12]의 처리 속도와 성능을 향상시켰고, 이듬해 라이언 달(Ryan Dahl)[5]은 자바스크립트 기반의 서버 사이드(server side) 언어

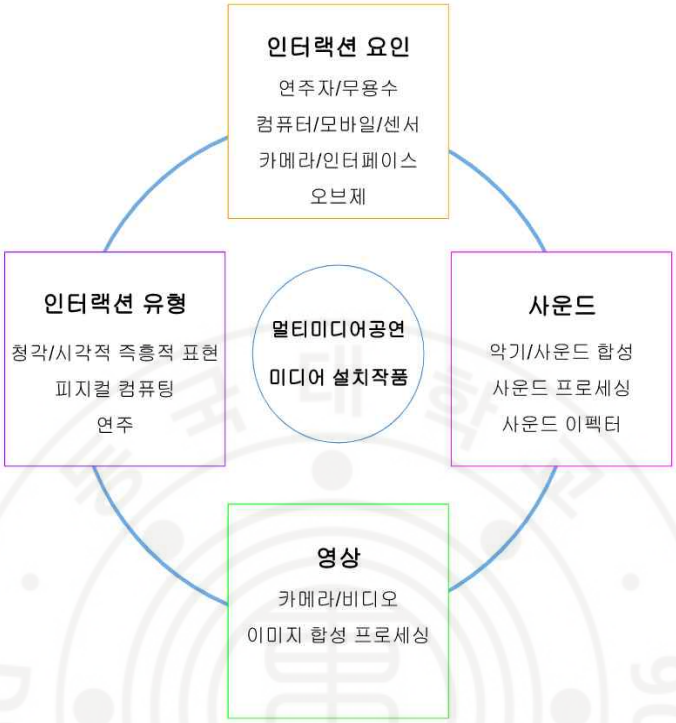
3) 웹 표준 기술을 사용하여 각 브라우저에서 동일한 웹을 구현하는 기술.
4) 월드 와이드 웹의 표준을 제안하고 연구 하는 단체.
5) 1980 ~, 미국 출신의 소프트웨어 엔지니어.

Node.js를 개발하면서 비동기(asynchronous) 방식의 인터랙티브(interactive) 네트워킹 플랫폼을 제공하였다 [13][14].

위와 같은 웹의 발전은 다음과 같은 기술과 융합되어 실시간으로 음악과 영상, 텍스트 등의 다양한 인터랙티브 멀티미디어 시스템을 제어하는 웹 기반의 인터랙티브 응용 애플리케이션 구현을 가능하게 하였다.

- 모바일과 PC를 통한 접근성과 확장성
- 3G, 4G, 5G, Wi-Fi 기술의 발달로 광범위한 인터넷 접속 환경
- 자바스크립트기반의 서버 구축 및 비동기 네트워킹 통신
- HTML5와 자바스크립트 API를 활용한 오디오, 그래픽 제어
- 모바일과 PC의 기본 내장 센서와 마이크, 카메라 활용

이러한 기술을 통해 웹 기반의 인터랙티브 멀티미디어 시스템에 대한 관심은 점차 증대되어 다양한 연구와 개발이 진행되고 있다. 이에 웹 기반의 멀티미디어 기술 활용을 위해 현재 멀티미디어 시스템에 사용되는 설치형(installation) 애플리케이션의 특징을 살펴보고자 한다. 최근까지 연구되고 있는 설치형 애플리케이션 기반의 인터랙티브 멀티미디어 시스템은 [그림-1.1]과 같은 특징을 가지고 있다.



[그림-1.1] 인터랙티브 멀티미디어 시스템 특징

위에서 제시된 특징들은 공연이나 설치작품의 컨셉에 따라 다양한 형태로 융합되어 인터랙티브 멀티미디어 시스템을 구현한다.

2010년 11월, 인터랙티브 멀티미디어 공연 "보는소리 듣는영상(Seeing Sound Listening Image) Ⅶ(이해랑극장)"에서 초연된 "혼(魂)"[6]은 색소폰의 음색분석을 통하여 사운드를 프로세싱하고 실시간으로 영상을 제어하는 인터랙티브 작품이다. 시스템 구현을 위해 설치형 애플리케이션인 Max/MSP[7]와 Quartz Composer[8]를 사용하였다. 특히 Max/MSP는 다양

6) 본 연구자가 구현한 멀티미디어 작품, https://youtu.be/vKvjo14Tm-U.
7) Cycling'74사가 개발한 멀티미디어 응용 프로그램.
8) Apple사에서 제공하는 비주얼 프로그램.

한 사운드 제어와 실시간 인터랙션, 네트워킹 등의 환경을 제공하고 있어 인터랙티브 멀티미디어 시스템을 구현하기에 최적화되어있다. 하지만 설치형 애플리케이션으로 구현된 시스템은 다음과 같은 이슈를 제공하면서 공연 재연과 시스템 활용에 비효율적인 문제점을 드러냈다.

첫째, 공연 시스템에 필요한 컴퓨터, 컨트롤러 등 장비에 대한 휴대성과 이동성이 제한적이다.

둘째, 시스템 환경 구축을 위한 시간이 많이 소요된다. 시간적 제한을 가진 공연장에서 시스템 구축을 위한 시간활용이 비효율적이다.

셋째, 실시간 인터랙션 기술을 사용할 경우 네트워킹 설정을 필요로 한다.

넷째, 서로 다른 운영체제에서 애플리케이션 호환이 불가하며 애플리케이션 버전에 따른 소스 호환에 버그를 발생시키기도 한다.

이에 따라 접근성과 호환성이 좋은 HTML5와 웹 기술을 활용하여 인터랙티브 멀티미디어 시스템을 구현한다면 앞서 논의된 공연 재연과 시스템 활용에 대한 효율성이 증대될 것이라는 가능성을 제언하고자 한다. 웹 기반의 인터랙티브 멀티미디어 시스템은 추가 장비 없이 PC와 모바일을 통해 멀티미디어 시스템 구축이 가능하다. 웹 애플리케이션은 웹 호스팅, 서버 호스팅 등의 무료로 제공하는 서비스가 많아 큰 비용 없이 시스템 구현이 가능하며 URL을 통해 실시간 인터랙션이 가능하다. 또한 HTML5, 자바스크립트 등의 다양한 웹 기술을 활용할 수 있어 확장성이 좋고 오프 소스 기반의 활발한 커뮤니티가 활성화되어 기술 개발, 디버깅(debugging) 등의 정보 공유가 가능하다. 이러한 특징을 통해 웹 기반

의 공연 시스템 구축 환경 및 활용에 대한 효율성을 검증하고자 한다.

본 논문에서는 이를 위해 웹 기술을 활용하여 공연에 필요한 인터랙티브 웹 애플리케이션을 구현하고자 한다.

본 논문의 목적은 이상의 논의를 중심으로 크게 두 가지로 구분된다.

첫 번째, 사용자 측면에서 공연에 필요한 인터랙티브 웹 애플리케이션 구현을 위해 웹에서 제공하는 오디오 기술을 기반으로 오디오 라이브러리를 설계하고 구현하여 기술적 특징과 적용방법을 제시한다.

두 번째, 제안된 웹 오디오 라이브러리의 적용방법을 중심으로 공연 이외에도 다양한 인터랙티브 웹 애플리케이션에서 활용될 수 있는 접근 방법을 고찰하고 시스템을 구현한다.

이를 위해 웹의 기술적 배경을 살펴보고 기존에 구현된 오디오 라이브러리를 비교분석하여 본 연구에 최적화된 라이브러리를 구현한다. 또한 라이브러리를 활용해 구현된 인터랙티브 웹 애플리케이션의 성능을 분석하고 향후 연구방향을 제시한다.

2. 연구 범위 및 구성

현재 끊임없이 발전하고 있는 자바스크립트와 HTML5, Web Audio API를 중심으로 웹 기술의 장점을 활용하고 단점을 보완하여 다음과 같은 주제로 연구를 진행하였다.

첫 번째는 웹 오디오 라이브러리 개발이며 연구 목적은 최대한 빠르고 쉽게 오디오 공연 시스템을 개발 할 수 있도록 오디오 라이브러리를 제공하는 것이다. 또한 효과적인 라이브러리 활용을 위해 튜토리얼 문서와 데모 애플리케이션을 제공하고 라이브러리 테스트와 디버깅(debugging)에 필요한 웹 오디오 에디터를 구현한다.

두 번째는 인터랙티브 공연을 위한 오디오 시스템 개발이다. 웹 오디오 라이브러리를 활용하여 공연에 필요한 마이크 입력, 사운드 이펙터, 이퀄라이저, 딜레이 등의 오디오 컴포넌트를 구현한다. 실시간 네트워크 통신을 위해 서버사이드 언어인 Node.js를 사용하여 메인 애플리케이션과 컨트롤 애플리케이션간의 실시간 오디오 인터랙션 구현을 목적으로 한다.

세 번째는 개발된 웹 오디오 라이브러리와 네트워킹 기술을 활용한 인터랙티브 미디어 설치 작품 개발이다. 실시간 네트워크 통신을 위해 Node.js를 사용하여 미디어 설치 작품을 개발한다.

본 논문은 연구 주제에 따라 아래와 같이 여섯 장으로 구성된다.

1장은 연구 배경과 목적, 범위 그리고 논문 구성에 대해 살펴본다.

2장에서는 관련 연구를 기술한다. 먼저 Web Audio API의 특징에 대

해 살펴보고, 그 다음 웹 기반의 오디오 라이브러리와 인터랙티브 시스템의 특징에 대해 고찰한다.

3장은 본 논문의 중요 기술인 웹 기반의 오디오 라이브러리에 관해 기술한다. 라이브러리의 아키텍처와 특징을 살펴보고 적용 방법을 제공하며 라이브러리 활용을 위해 구현된 웹 오디오 에디터에 대해 살펴본다.

4장은 오디오 라이브러리를 활용하여 구현된 인터랙티브 웹 애플리케이션인 오디오 시스템과 미디어 설치작품의 구조와 적용된 기술에 관하여 알아본다.

5장은 구현된 오디오 라이브러리와 웹 애플리케이션의 성능을 고찰하고 기존의 연구와 비교분석한다.

6장에서는 결론을 도출하고 구현된 웹 오디오 라이브러리와 애플리케이션의 연구 방안을 제안한다. 또한 참고문헌과 본 연구의 오디오 라이브러리와 웹 애플리케이션 주소, 소스 코드 정보를 마지막으로 본 논문을 마무리한다.

Ⅱ. 관련 연구

1. Web Audio API

2011년 구글에서 개발한 Web Audio API는 실시간 오디오 처리, 사운드 합성, 이펙터, 게임 음악 등 전문적이고 복잡한 응용 애플리케이션 개발을 위해 구현된 API로 웹 기반의 오디오 시스템 구현에 새로운 가능성을 제시하였다 [27][28]. 이미 HTML5에서도 스트리밍을 통한 오디오 재생 기능을 제공하고 있지만, 정밀한 오디오 시스템 구현에는 한계가 있다. 이러한 대안으로 구현된 Web Audio API는 C++[9] 언어로 개발되었고 자바스크립트 문법을 사용하여 브라우저에서 실행되는 환경을 제공한다.

Web Audio API가 제공하는 오디오 인터페이스는 오디오 노드(Audio Node)를 통해 오디오 인터페이스의 입출력을 연결한다. 이러한 연결 구조를 모듈형 라우팅이라 정의하고 있다 [27][28]. 오디오 노드는 다수의 입력과 출력이 가능하고 모듈형 라우팅 구조로 오디오를 실행시키며, 오실레이터(oscillator), 필터(filter), 패너(panner) 및 컨볼루션(convolution), 오디오 인풋 등으로 구성되어 있다. 또한 W3C는 XMLHttpRequest[10], 캔버스 2D[11] 및 WebGL 3D 그래픽 API [31]와 함께 Web Audio API

9) C 언어의 문법에 객체 지향 프로그래밍 개념이 추가된 언어.
10) 서버통신을 위해 Ajax 프로그래밍에 주로 사용되는 객체.

사용을 권장하고 있다.

Web Audio API의 주요 특징은 자바스크립트 기반으로 다양한 패키지와 모듈 활용이 가능하며 HTML5가 제공하는 API와 호환성이 좋다. 또한 오픈소스 기반의 웹 기술과 다양한 개발 환경을 제공한다. 하지만 오디오와 비디오 스트리밍을 사용할 경우 보안 정책으로 인해 HTTPS (Hypertext Transfer Protocol over Secure Socket Layer)[12]를 사용해야 하며 특정 기능이 일부 브라우저에서 호환되지 않아 앞으로 개선해야 할 문제가 남아있다.

11) HTML5에서 지원하는 2D 그래픽.
12) 웹 통신 프로토콜인 HTTP에 보안을 위해 인증과 암호화를 추가한 프로토콜.

2. 웹 오디오 라이브러리

Web Audio API는 실시간 오디오처리와 소리 합성 등의 기능을 제공하지만, 난이도 있는 자바스크립트 문법으로 설계되어 애플리케이션 초기 구현의 어려움이 있다. 하지만 관련 라이브러리들이 개발되면서 보다 빠르고 간단하게 애플리케이션 구현이 가능해졌으며, 크게 두 가지 형태로 발전해 왔다. 첫 번째는 오디오 처리 환경만 제공하는 형태이고 두 번째는 오디오 처리 환경과 애플리케이션구축에 필요한 비주얼라이제이션(visualization)과 UI(User Interface) 등의 플랫폼(platform)을 제공하는 형태이다. 이러한 발전 형태를 중심으로 기존의 주요 라이브러리에 대해 살펴본다.

Audiolib.js는 Web Audio API로 구현된 최초의 오디오 라이브러리로 2012년 Rovio Entertainment[13]사의 Jussi Kalliokoski에 의해 개발되었으며 오디오 이펙터와 사운드 합성, 샘플러 등의 다양한 오디오 플랫폼을 제공한다. 또한 Web Audio API가 제공하는 오디오 기술을 효과적으로 활용하여 이후에 구현된 라이브러리 개발에 큰 영향을 주었다 [32].

가장 큰 영향을 받은 라이브러리는 캘리포니아 대학 샌타바버라(Santa Barbara) Media Arts & Technology 연구소의 Charles Roberts와 JoAnn Kuchera-Morin이 개발한 Gibber이다. Audiolib.js와 같은 해에 개발된 Gibber는 Audiolib.js의 오디오 플랫폼 기반으로 개발되어 다양한 오디오 기술을 사용할 수 있고 실시간 라이브 코딩 환경과 간결한 코드

13) 2003년에 설립된 핀란드의 모바일 게임회사.

신택스(syntax)[14]를 제공한다 [33]. 간결한 코드 신택스는 짧은 코드 양과 가독성이 좋은 장점을 가지고 있고 빠른 개발 환경을 제공하기 때문에 라이브러리 개발에 있어 반드시 고려해야 하는 사항이다.

[그림-2.1]의 (a)는 Audiolib.js의 코드 신택스이고 (b)는 Gibber의 코드 신택스로 (b)가 (a)에 비해 간결해 가독성이 좋다.

(a)

```
// Audiolib.js
var sound =new audioLib.Oscillator(4410, 440);
sound.waveShape ='square';
```

(b)

```
// Gibber
s = Square(440);
```

[그림-2.1] Audiolib.js와 Gibber의 코드 신택스

같은 해 Gibber와 다른 접근방법으로 개발된 Tuna는 DinahMoe AB[15] 사의 Oskar Eriksson이 개발한 기타(guitar) 이펙터 중심의 라이브러리로 앞서 연구된 라이브러리와 달리 오버드라이브(overdrive), 필터(filter), 딜레이(delay) 등의 오디오 이펙터를 제공한다 [34]. 또한 Tuna는 앞서 연구된 라이브러리에 비해 이펙터 제공에만 중점을 두어 가볍고 빠르다

14) 프로그래밍 언어의 코드 형태, 구조, 문법.
15) 2008년에 설립된 스웨덴의 음악 및 사운드 제작 회사.

는 장점이 있지만 애플리케이션 구현을 위해 추가 기능을 필요로 한다.

이후 UI 플랫폼과 비주얼라이제이션 등의 개발에 필요한 추가기능을 제공하는 라이브러리들이 발전하기 시작하였다.

WAAX(Web Audio API eXtension)는 2013년 스탠포드 대학 CCRMA (Center for Computer Research in Music and Acoustics)연구소의 최홍찬과 Jonathan Berger가 연구한 Web Audio API 기반의 통합 오디오 개발환경이다. WAAX는 Web Audio API 사용을 위해 브라우저간의 호환성 이슈의 대안을 제시하고 오디오 플러그인과 데이터 제어를 위한 UI 플랫폼과 비주얼라이제이션을 제공한다. 또한 간결한 코드 신택스를 통해 사용자들은 빠르게 오디오 시스템을 구현할 수 있다 [35][36].

WAAX와 같은 해 개발된 Gibberish.js와 Interface.js는 캘리포니아 대학 Media Arts and Technology 연구소의 Charles Roberts와 Matthew Wright, KAIST Graduate School of Culture Technology의 Graham Wakefield가 연구한 라이브러리이다. Gibberish.js는 오디오 DSP 라이브러리로 다양한 오실레이터와 폴리포닉 신스(polyphonic synths), 이펙터, 필터, 퍼커션, 등으로 구성되어 오디오 시스템 환경을 제공한다. 퍼커션의 경우 TR-808 기능을 Web Audio API로 재현한 특징을 가지고 있다 [37]. Interface.js는 마우스나 화면 터치 이벤트를 통해 Gibberish.js의 오디오를 GUI로 제어하는 라이브러리이며 실시간으로 MIDI와 OSC 데이터 통신 기능을 제공한다 [37]. 이와 같은 Gibberish.js의 오디오 시스템과 Interface.js의 GUI 기능을 잘 활용한다면 빠르고 쉽게 오디오 애플리케이션을 구현할 수 있다.

지금까지 기존의 웹 기반의 오디오 라이브러리에 대해 살펴보았다. 본 논문은 이미 연구된 라이브러리들의 특징을 기반으로 다음과 같은 접근 방법을 제시한다. 첫째, 간결한 코드 신택스를 통해 빠르게 오디오 애플리케이션 구현이 가능한 최소한의 개발 환경을 제공한다. 둘째, 오디오 공연 시스템 이외의 애플리케이션 구현에 필요한 유틸리티를 제공하여 추가 프로그래밍을 최소화한다. 또한 라이브러리 사용을 위한 파일 설치와 초기 환경설정을 단순화시켜 효율성을 증대한다.

3. 인터랙티브 시스템

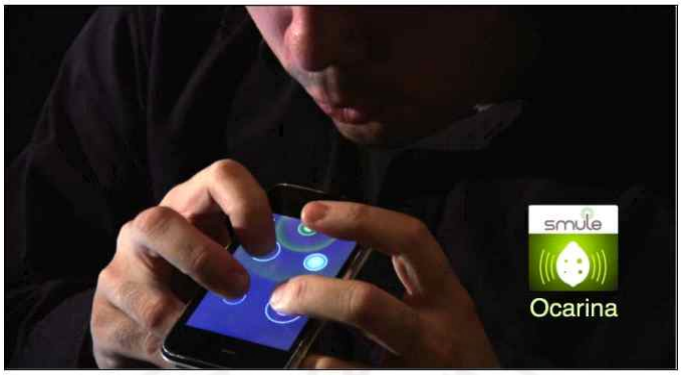
네트워킹 통신 기술의 발전은 미디어 아트, 설치작품, 멀티미디어 공연 등에 실시간 인터랙션 시스템 구현을 가능하게 하였다. 본 장에서는 인터랙티브 웹 애플리케이션 개발을 위해 인터랙티브 기술을 활용한 오디오 시스템과 미디어 설치작품에 관한 관련 연구를 살펴보겠다.

Camus2는 모바일의 카메라를 활용한 협업(collaborative) 음악 프로젝트로 2007년 베를린 대학교 Deutsche Telekom Laboratories 연구소의 Michael Rohs와 Georg Essl가 개발하였다. 모바일의 카메라를 이용해 위치와 높이 등을 트래킹 하고 블루투스(Bluetooth)[16]를 통해 트래킹 데이터를 실시간으로 컴퓨터에 전송한다. 컴퓨터로 전송된 데이터는 미디 포맷으로 변환되어 음악 제어에 활용된다 [70]. 본 작품에 사용된 모바일과 컴퓨터는 설치형 애플리케이션으로 구현되었다. 모바일의 경우 가속도센서, 위치센서, 터치센서, 카메라, 마이크 등의 기능을 활용할 수 있어 인터랙션 시스템에 활용도가 높다. Camus2는 이러한 모바일의 카메라를 통해 입력되는 데이터로 음악을 제어하는데 활용하였다.

Camus2와 모바일 접근방법이 유사한 오카리나(Ocarina)는 iOS 플랫폼의 악기 애플리케이션으로 2009년 스탠포드 대학 CCRMA연구소의 Ge Wang이 개발하였다. 오카리나는 척(chuck) [57]을 사용하여 사운드를 구현하였다. 사운드를 컨트롤하기 위해 모바일의 마이크와 멀티 터치 센서, 가속도계를 이용하였다. 또한 오카리나에 접속한 정보를 서버로 전송하여 사용자의 위치와 오카리나 사용자수를 제공한다 [69].

---

16) 2.42～485GHz 대역의 주파수를 이용한 근거리 데이터 통신방식.



[그림-2.2] 모바일 오카리나 [69]

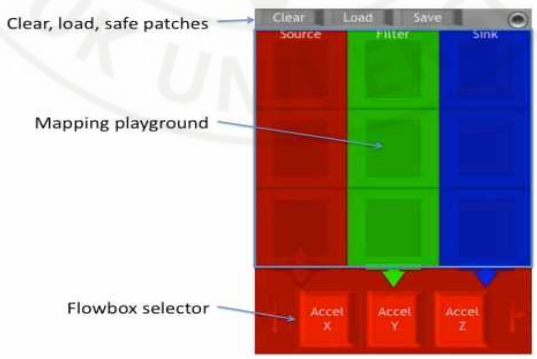
오카리나 애플리케이션을 개발한 Ge Wang은 2010년 같은 연구소의 Nicholas J. Bryan 외 2명과 MoMu 애플리케이션을 개발하였다.

MoMu는 iOS 플랫폼기반의 모바일 뮤직 애플리케이션이다. MoMu는 모바일의 가속센서와 GPS(Global Positioning System)[17]를 활용하여 오디오와 그래픽 등을 제공한다 [68]. 같은 해 Ge Wang은 2010년 같은 연구소의 Jieun Oh 외 3명과 스탠포드 모바일 폰 오케스트라(MoPhO) 프로젝트를 구현하였다. 스탠포드 모바일 폰 오케스트라(MoPhO)는 스탠포드의 랩탑(laptop) 오케스트라를 배경으로 발전하였다. MoPhO는 모바일의 이동성과 접근성을 활용한 인터랙션과 새로운 플랫폼을 제공한다 [71]. Ge Wang은 모바일을 활용한 인터랙션과 오디오 접근의 다양한 가능성을 제시하였다.

17) 위성에서 보내는 신호를 통해 현재 위치를 제공하는 시스템.

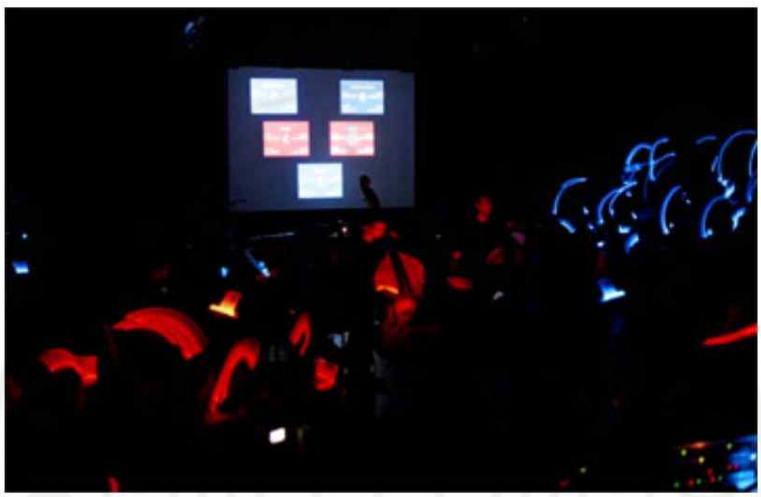
[그림-2.3] 모바일 폰 오케스트라 (MoPhO) [71]

urMus는 모바일의 멀티 터치기술을 활용한 음악 악기 플랫폼으로 2010년 미시간 대학 Electrical Engineering & Computer Scienceand Music 연구소의 Georg Essl과 베를린 공대 Deutsche Telekom Laboratories 연구소의 Alexander Muller가 개발한 설치형 애플리케이션이다. Lua 스크립트 언어로 구현되었으며 애플리케이션의 UI를 맵핑하여 오디오와 미디어간의 인터랙션을 실행한다 [60].



[그림-2.4] urMus 모바일 인터페이스 [60]

Glimmer는 2005년 Georgia Institute of Technology Music Department의 Jason Freeman이 구현한 공연으로 오케스트라와 관객 그리고 카메라 기술을 활용한 관객참여 프로젝트이다.



[그림-2.5] Glimmer 공연 프로젝트 [75]

공연에 활용된 카메라는 Max/MSP를 통해 관객들의 형광봉 움직임을 트레킹 한다. 관객들은 형광봉의 움직임을 제어하고 이를 통해 오케스트라와 인터랙션 공연을 만들어간다 [75].

Max/MSP를 활용한 Glimmer와 달리 다양한 프로그래밍 언어를 사용한 TweetDreams는 트위터(Twitter)를 활용한 관객참여 프로젝트이다. 2011년 스탠포드 대학 CCRMA연구소의 Luke Dahl, Jorge Herrera, Carr Wilkerson이 개발하였다. TweetDreams는 파이썬(Python) [56]을 사용하여 검색어를 처리하고 척을 사용하여 오디오를 제어한다. 또한 프로세싱(Processing) [58]을 사용하여 비주얼을 표현한다 [65].

massMobile은 관객이 모바일을 사용하여 라이브 음악 퍼포먼스에 참여하는 시스템으로 2012년 Georgia Tech Center for Music Technology의 Nathan Weitzner와 그 외 연구자 3명이 개발하였다. massMobile은 Max/MSP 기반으로 설계되었고, 실시간 양방향 통신을 통해 시스템을 구현한다. massMobile은 관객이 모바일을 통해 공연의 조명을 제어하거나 설정된 음악 시퀀서를 제어하는 방식을 제공한다. 하지만 퍼포먼스에 스토리가 있어 관객들의 학습을 필요로 한다 [62].

[그림-2.6] massMobile 애플리케이션 [62]

massMobile와 유사한 모바일 접근방법으로 개발된 echobo는 모바일을 음악 악기로 활용하여 관객이 공연에 참여하는 대규모 음악 프로젝트이다. 2013년 미시간 대학 컴퓨터 공학과의 Sang Won Lee와 Georgia Institute of Technology의 Center for Music Technology연구소 Jason Freeman이 구현하였다. echobo는 공연에 참여한 관객들 간의 실시간 인터랙션을 목적으로 하고 있다. 하지만 관객이 echobo에 참여하기 위해서는 애플리케이션 설치가 필요해 접근성이 웹에 비해 좋지 않다 [63].

[그림-2.7] 모바일기반의 echobo 프로젝트 [63]

SWARMED는 HTML5의 캔버스 기술을 활용하여 구현된 인터랙티브 모바일 관객 참여 시스템으로 2013년 앨버타 대학 컴퓨터 공학과 Abram Hindle이 개발하였다. 관객들이 모바일을 통해 실시간으로 UI를 제어하면 네트워킹을 통해 사운드가 재생하는 작품이다. XMLhttpRequest 기술을 활용하여 인터랙션 네트워킹 시스템을 구현하지만 동기방식의 통신을 지원하고 있어 실시간 인터랙션에 시간차를 발생시킨다 [59]. 이에 본 연구에서는 비동기 방식의 Node.js를 사용하여 실시간 인터랙션의 시간차를 최소화한다.

본 논문은 위의 연구사례들을 중심으로 웹 오디오 라이브러리를 개발하고 이를 통해 인터랙티브 웹 애플리케이션을 구현하고자 한다. 다음 장에서는 웹 오디오 라이브러리를 정의하고, 라이브러리의 구조와 특징 그리고 적용 방법에 대해서 알아본다.

Ⅲ. 웹 오디오 라이브러리 개발

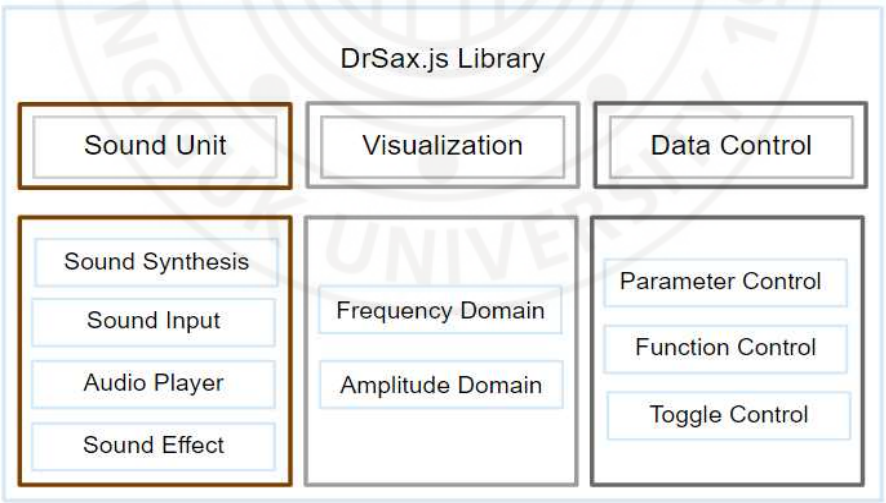
DrSax.js는 Web Audio API 기반의 웹 오디오 라이브러리로 인터랙티브 멀티미디어 공연에 사용 가능한 웹 오디오 애플리케이션 구현을 위해 개발되었다. 자바스크립트 문법과 오디오 프로세싱에 익숙하지 않은 사용자에게 오디오 합성과 프로세싱, 사운드 이펙터, 오디오 파일 재생 등의 인터페이스를 간결한 코드 신택스로 제공한다. 여기서 인터페이스는 DrSax.js를 구성하는 요소 중 속성 값이나 메서드(method)를 가지고 독립적인 기능을 제공하는 함수로 정의한다. 이러한 인터페이스는 쉽고 빠르게 애플리케이션 구현을 가능하게 하며 공연 시스템 이외에 미디어 설치 작품이나 다른 오디오 시스템에 활용이 가능하다.

본 장에서는 DrSax.js의 아키텍처와 특징, 구성요소에 대해 설명한다. 그리고 애플리케이션 개발을 위해 라이브러리를 적용하는 방법과 구현 모델을 제시한다. 또한 애플리케이션 개발에 필요한 라이브러리 코드 디버깅 및 에디팅(editing)을 위해 웹 오디오 에디터를 구현한다.

1. DrSax.js 아키텍처 및 특징

DrSax.js는 오디오 애플리케이션 구현을 위해 3가지 구성요소 Sound Unit, Visualization, Data Control로 설계되었다. Sound Unit은 실시간 오디오 처리와 사운드 합성 등의 오디오 시스템을 구성할 수 있는 오디오 인터페이스로 구성된 코어 엔진이며 Visualization은 Sound Unit의 출력 데이터를 비주얼라이제이션 하는 인터페이스로 구성되어 있다. Data Control은 Sound Unit 인터페이스의 파라미터 값을 변화시키는 인터페이스로 구성되어 있고 웹 브라우저의 UI를 통해 작동한다.

Visualization과 Data Control은 오디오 시스템 이외의 애플리케이션 구현에 필요한 기능을 제공하는 유틸리티 인터페이스이다.



[그림-3.1] DrSax.js 라이브러리 구조

[그림-3.1]은 DrSax.js 라이브러리 구조를 나타내고 있으며 각 구성요소의 특징은 다음과 같다.

- **Sound Unit**

오디오 시스템 구현에 필요한 사운드 합성, 오디오 파일 재생, 사운드 인풋, 사운드 이펙터 등의 오디오 인터페이스로 구성되어 있다.

- **Visualization**

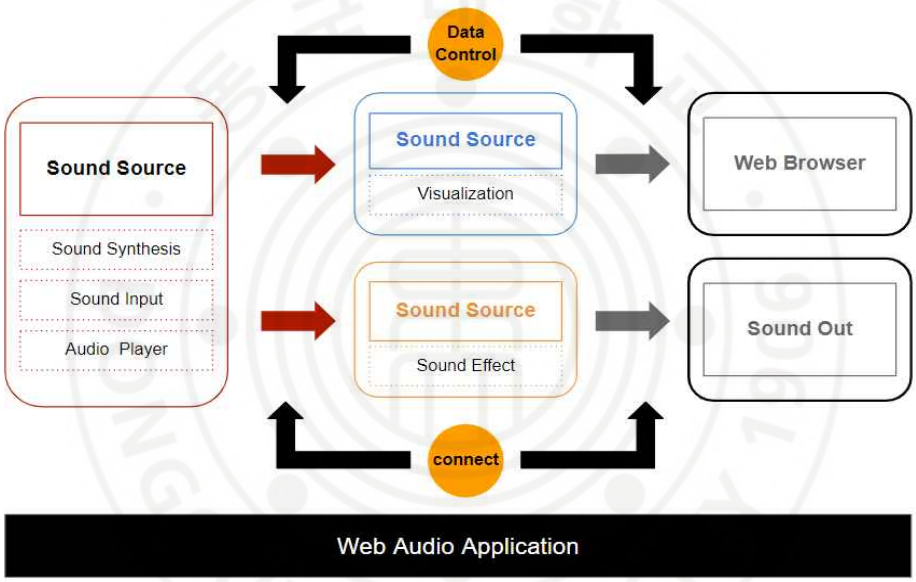
Sound Unit의 구성요소 중 출력 값을 가진 오디오 인터페이스의 출력 데이터를 분석하여 프리컨시 도메인(frequency domain)과 앰플리튜드 도메인(amplitude domain)으로 비주얼라이제이션 하는 인터페이스를 제공하며 HTML5의 캔버스 API를 통해 화면에 그려진다.

- **Data Control**

Sound Unit이 제공하는 오디오 인터페이스의 파라미터 값 제어와 인터페이스나 오디오 시스템의 on/off 설정을 위해 제공되는 콘트롤러 함수이다.

또한 DrSax.js의 구성요소는 [그림-3.2]의 프로세스를 통해 오디오 애플리케이션이 구현되도록 설계되었다.

Sound Source의 출력 값은 Visualization과 연결되어 UI 화면에 그려지고 Data Control은 Sound Unit의 인터페이스 파라미터 값을 제어하여 사운드와 Visualization을 변화시킨다.



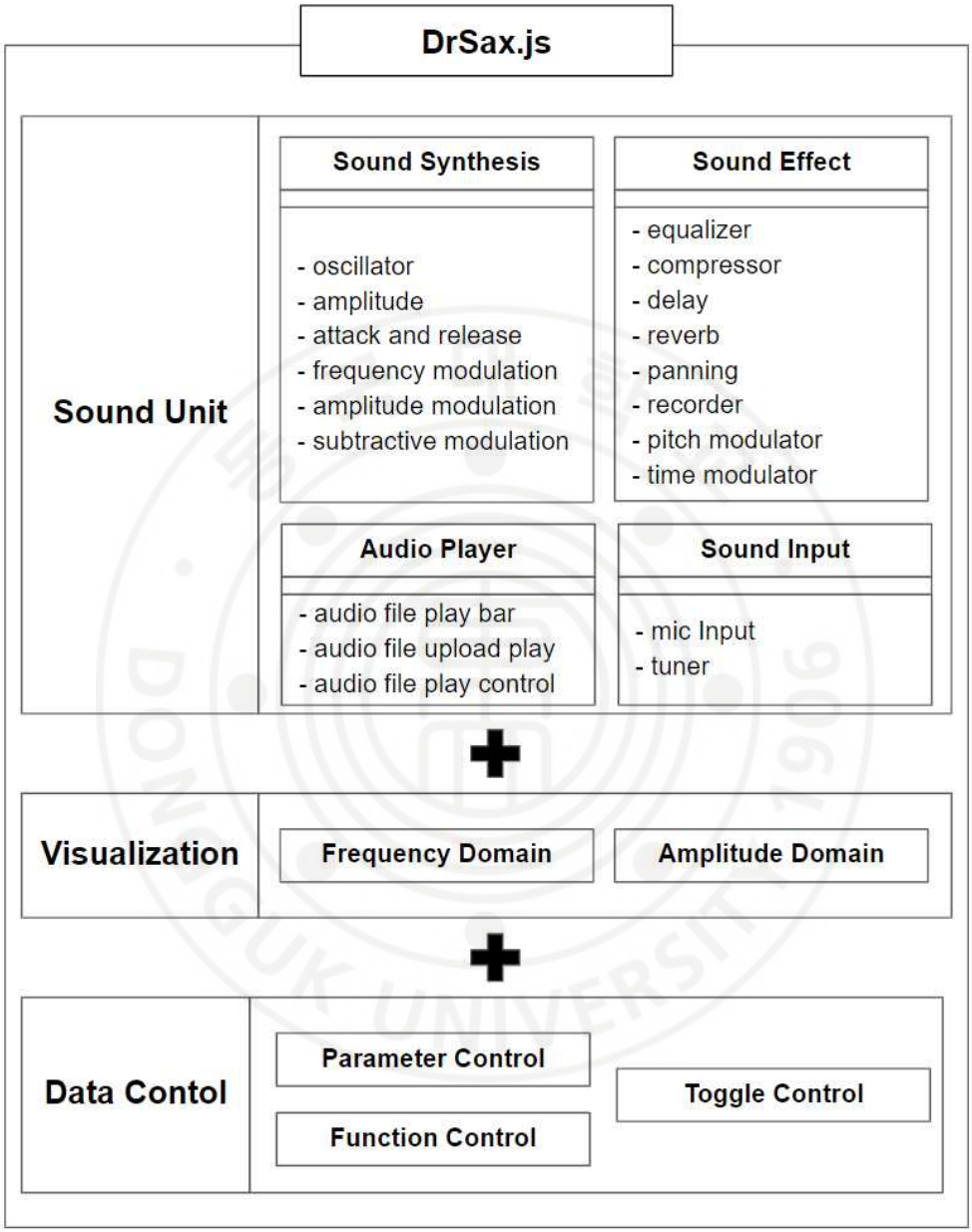
[그림-3.2] DrSax.js의 구성요소 프로세스

이와 같은 프로세스를 기반으로 오디오 시스템을 구성하면 비주얼라이제이션과 오디오 파라미터 제어를 위한 별도의 추가 프로그래밍 없이 빠르고 간단하게 애플리케이션을 구현할 수 있다. 버튼과 다이얼, 디자인 등의 애플리케이션 UI를 구현하기 위해서는 HTML5, CSS3 뿐만 아니라

웹 UI 프레임워크인 NexusUI 2.0, g200kg 등을 활용할 수 있다. 이러한 DrSax.js의 주요 특징은 다음과 같다.

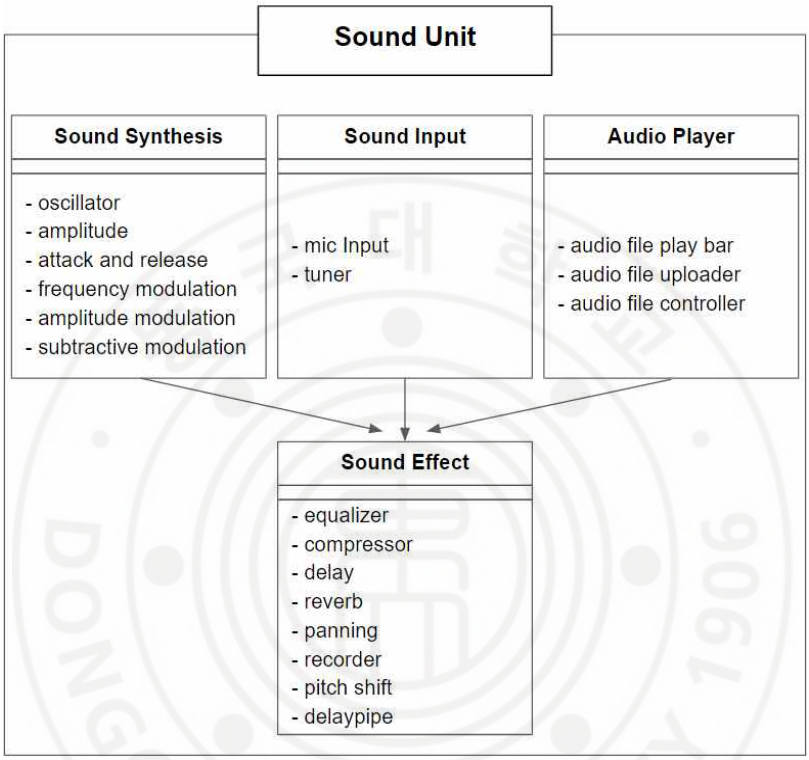
- 코딩 신택스를 최소화하여 가독성과 프로그래밍의 효율 증대
- 심플한 비주얼라이제이션 제공
- js 파일을 다운받아 바로 사용 가능
- 크로스 브라우징 호환성
- Web Audio API 기반의 오디오 프로세싱/이펙터
- 명료하고 간결한 코드로 오디오 이펙터와 사운드 프로세싱 시스템을 손쉽게 구현
- 오디오 데이터 처리를 간결하게 제어할 수 있는 데이터 컨트롤러 제공
- MIT 라이센스(Massachusetts Institute of Technology License)로 누구나 자유롭게 사용 가능

[그림-3.3]은 DrSax.js 라이브러리의 인터페이스 구성요소를 나타내고 있다. 다음 항에서는 DrSax.js 구성요소들의 종류와 특징에 대해 자세히 살펴보겠다.



[그림-3.3] DrSax.js의 인터페이스 구성요소

1) Sound Unit

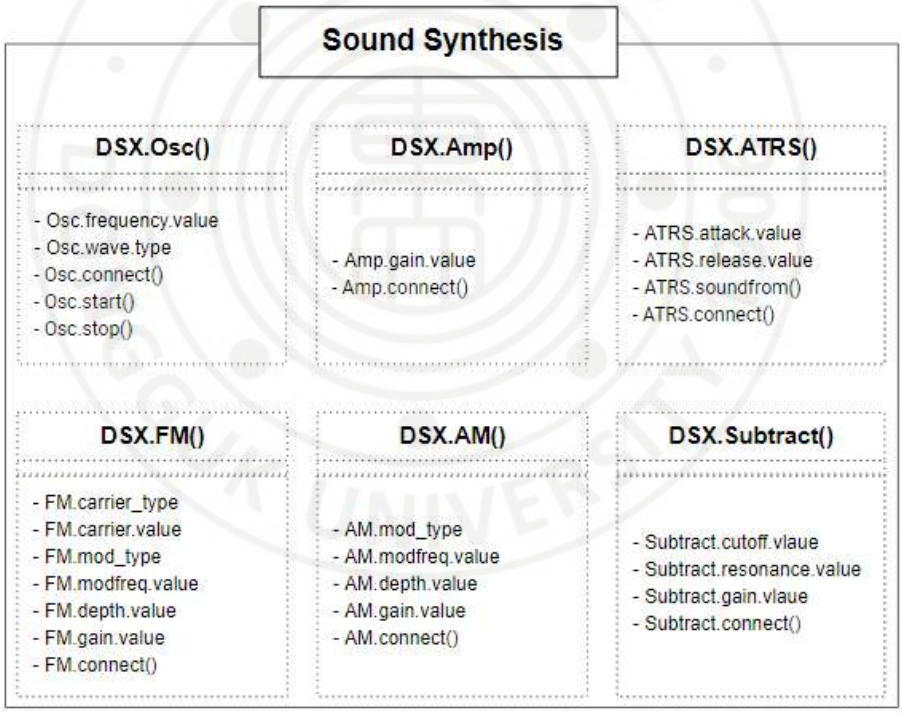


[그림-3.4] Sound Unit의 구성

Sound Unit은 DrSax.js의 핵심 기술인 오디오 인터페이스를 제공하는 코어 엔진으로 사운드를 출력하는 Sound Synthesis, Sound Input, Audio Player와 출력 오디오 데이터를 값을 받아 사운드를 변화시키는 Sound Effect로 구성되어 있다. Sound Unit의 구성요소에 대해 살펴보도록 하겠다.

● Sound Synthesis

Sound Synthesis는 웨이브(wave) 진동을 통해 소리를 발진시키는 oscillator와 음량을 제어하는 amplitude, 시간에 따라 음량 값을 제어하는 ADSR(Attack, Decay, Sustain, Release)의 어택(attack)과 릴리즈(release), 3개의 신디사이저 frequency modulation, amplitude modulation, subtractive modulation로 구성되어 있다.



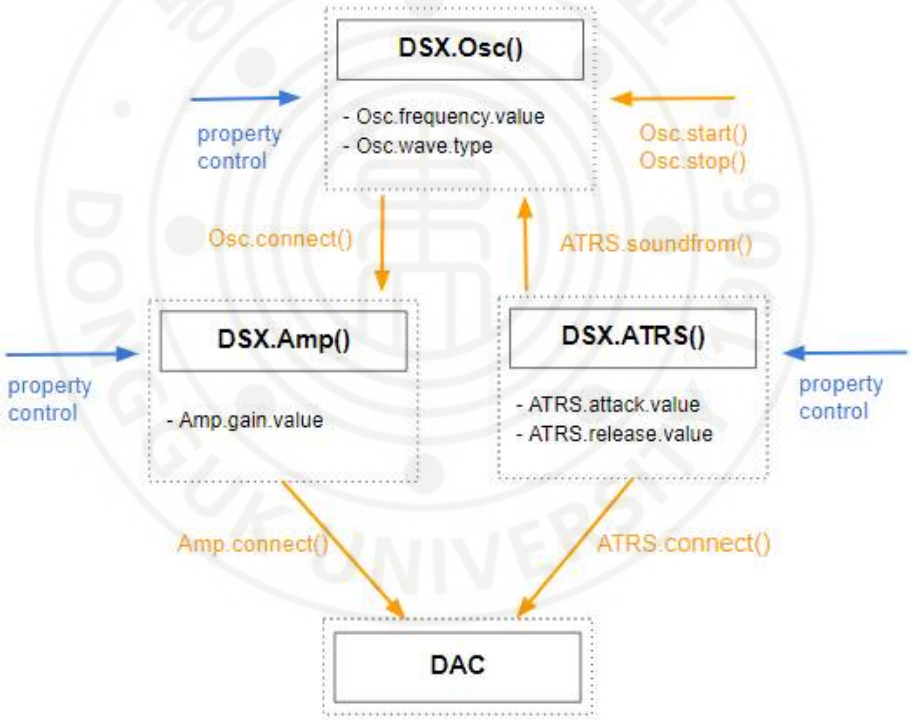
[그림-3.5] Sound Synthesis의 객체 구성요소

또한 oscillator와 3개의 신디사이저는 사운드 이펙터로 활용이 가능하여 다양한 사운드 효과를 제공한다. [그림-3.5]는 Sound Synthesis 객체 구성요소의 속성 값과 메서드를 나타내고 있으며 <표-3.1>은 각 객체의 기능에 대해서 설명하고 있다.

<표-3.1> Sound Synthesis 객체 종류와 기능

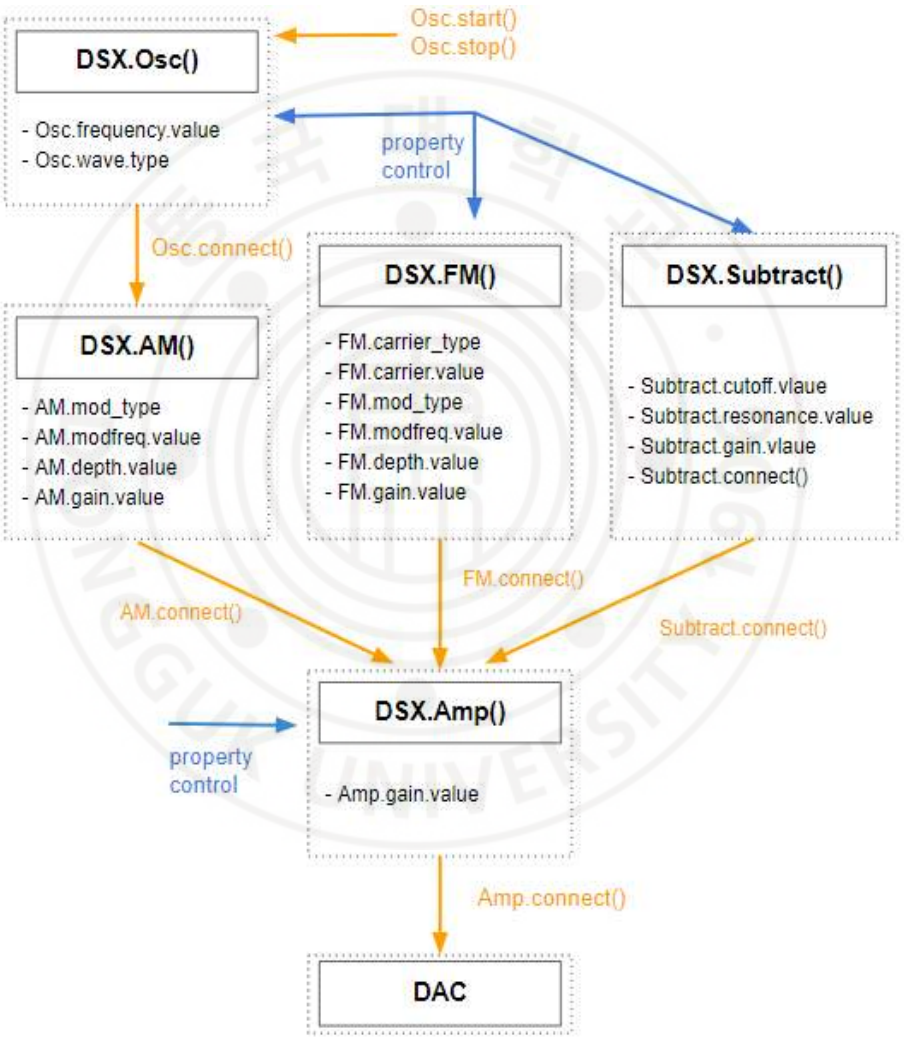
객체	기능
DSX.Osc()	프리컨시 값과 4가지 웨이브 파형을 통해 소리를 제공하는 oscillator 객체이다.
DSX.Amp()	오디오 인터페이스의 음량을 제어하고 AUX와 아웃풋 믹싱 등에 활용 가능한 amplitude 객체이다.
DSX.ATRS()	ADSR의 어택(attack)과 릴리즈(release)를 사용해 시간에 따른 음량 값을 제어하는 객체이다.
DSX.FM()	FM(Frequency Modulation) 신디사이저 객체이다.
DSX.AM()	AM(Amplitude Modulation) 신디사이저 객체이다.
DSX.Subtract()	Subtractive Modulation 신디사이저 객체이다.

[그림-3.6]은 Sound Synthesis의 DSX.Osc() 객체를 사용하여 오디오 시스템을 구현하는 라우팅 구조를 나타내고 있다. 기본적으로 사운드를 출력하는 객체는 음량을 제어하는 객체 DSX.Amp()와 연결된 후 DAC(Digital Audio Converter)로 연결되어 소리를 발생시킨다. DAC는 DrSax.js에서 최종 사운드 아웃 기능을 제공하는 객체이다.



[그림-3.6] DSX.Osc() 객체의 라우팅 흐름도

[그림-3.7]은 Sound Synthesis의 3가지 신디사이저 객체 DSX.FM(), DSX.AM(), DSX.Subtract()를 사용하여 오디오 시스템을 구현하는 라우팅 흐름도를 나타내고 있다.

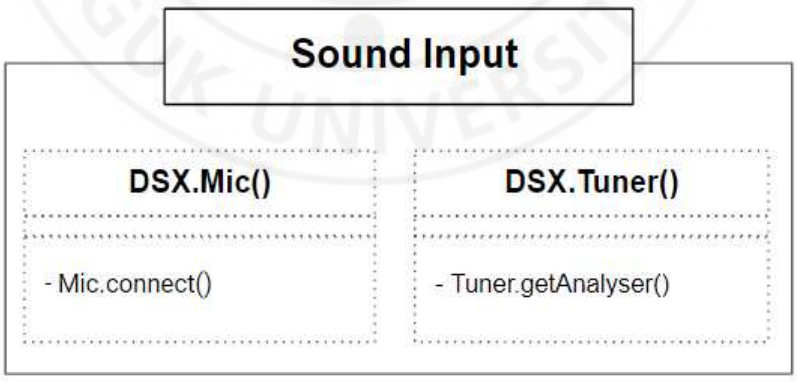


[그림-3.7] 신디사이저 라우팅 흐름도

● Sound Input

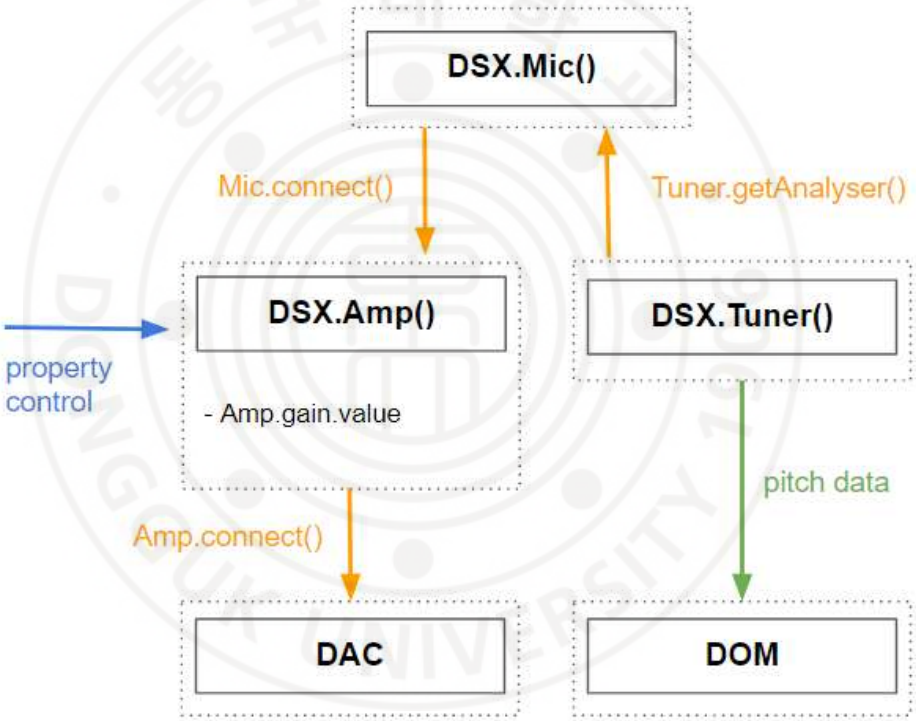
Sound Input은 웹 브라우저를 통해 사운드를 입력받는 마이크 인터페이스와 사운드 데이터를 분석하여 피치(pitch) 값을 제공하는 튜너(tuner) 인터페이스로 구성되어 있다.

마이크 인터페이스를 통해 전달되는 악기나 노래 등의 사운드 입력 값은 Sound Unit의 이펙터 인터페이스를 통해 프로세싱 되며 인풋 값을 입력받는 다른 인터페이스에 활용되기도 한다. 튜너 인터페이스는 마이크 인풋 값이나 출력 값을 갖는 오디오 인터페이스의 출력 데이터를 분석하여 피치로 제공한다. 튜너 인터페이스가 피치 값을 인지할 수 있는 마이크 인터페이스의 입력 값은 피치 값을 갖고 배경음악이 없는 단선율 멜로디만 가능하다. [그림-3.8]은 Sound Input 객체 구성요소의 메서드를 나타내고 있다.



[그림-3.8] Sound Input 객체 구성요소

[그림-3.9]는 Sound Input 객체의 라우팅 흐름도를 나타내고 있다. DSX.Mic() 객체는 기본적으로 DSX.Amp() 객체와 연결되어 소리를 발생시키고 DSX.Tuner() 객체는 DSX.Mic() 출력 값을 분석하여 웹 애플리케이션의 DOM(Document Object Model)[1]으로 데이터 결과를 전달한다.

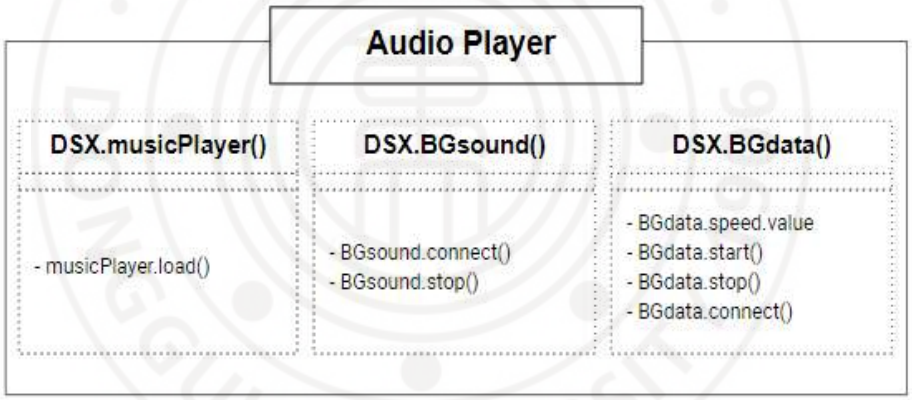


[그림-3.9] Sound Input 객체의 라우팅 흐름도

1) HTML의 문서를 표현하는 객체로 자바스크립트나 마크업을 통해 제어가 가능하다.

● Audio Player

Audio Player는 다음과 같은 3가지 종류의 오디오 파일재생 인터페이스를 제공한다. 첫 번째는 오디오 파일을 업로드 하여 재생하는 방식이고 두 번째는 HTML5의 오디오 태그를 이용하여 재생하는 방식이며 마지막은 URL 경로의 오디오 파일을 호출하는 방식이다. [그림-3.10]은 Audio Player 객체 구성요소의 속성 값과 메서드를 나타내고 있다.



[그림-3.10] Audio Player 객체 구성요소

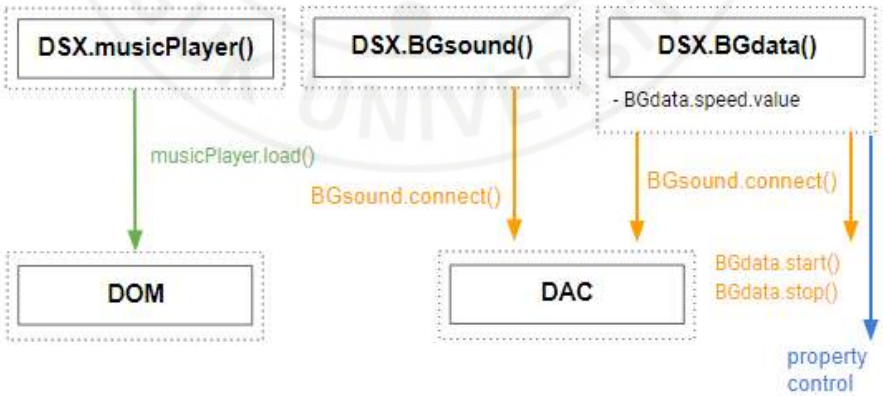
3가지 종류의 Audio Player는 공연 시스템의 MR(Music Recorded)[2] 반주 음악으로 사용되기도 하며 DSX.BGdata() 객체의 경우 음원 속도 변경이 가능하여 음원 소스의 코드나 멜로디 등의 화성 분석이나 공연 연습으로 활용이 가능하다.

2) 보컬이나 메인 멜로디악기를 제외한 배경음악.

<표-3.2> Audio Player 객체 종류와 기능

객체	기능
DSX.musicPlayer()	HTML에서 제공하는 인풋 태그와 오디오 태그를 사용하며 기본적인 재생, 일시정지 재생바(bar) UI를 제공한다.
DSX.BGsound()	HTML에서 제공하는 인풋 태그를 사용하며 메서드를 통해 재생, 정지 기능을 제공한다.
DSX.BGdata()	URL 경로의 오디오 파일을 재생하며 재생 속도를 제어하는 기능을 제공한다.

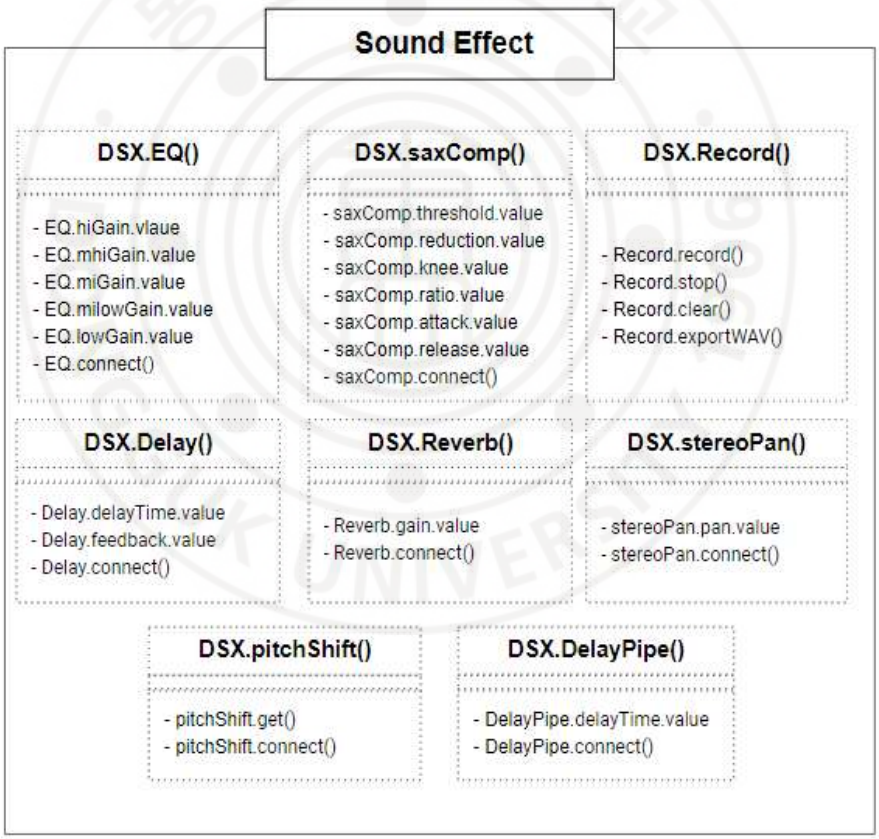
<표-3.2>는 Audio Player 객체 종류와 기능을 설명 있으며 [그림-3.11]은 Audio Player 객체의 라우팅 흐름도를 나타내고 있다.



[그림-3.11] Audio Player 객체의 라우팅 흐름도

● Sound Effect

Sound Effect는 일반적으로 많이 사용하는 equalizer와 compressor, stereo pan, delay, reverb, recorder와 공연을 위해 개발된 pitch modulator와 time modulator 이펙터를 제공하고 있다. 보다 쉽고 빠른 이펙터 사용을 위해 최소한의 파라미터 값을 설정하였다.



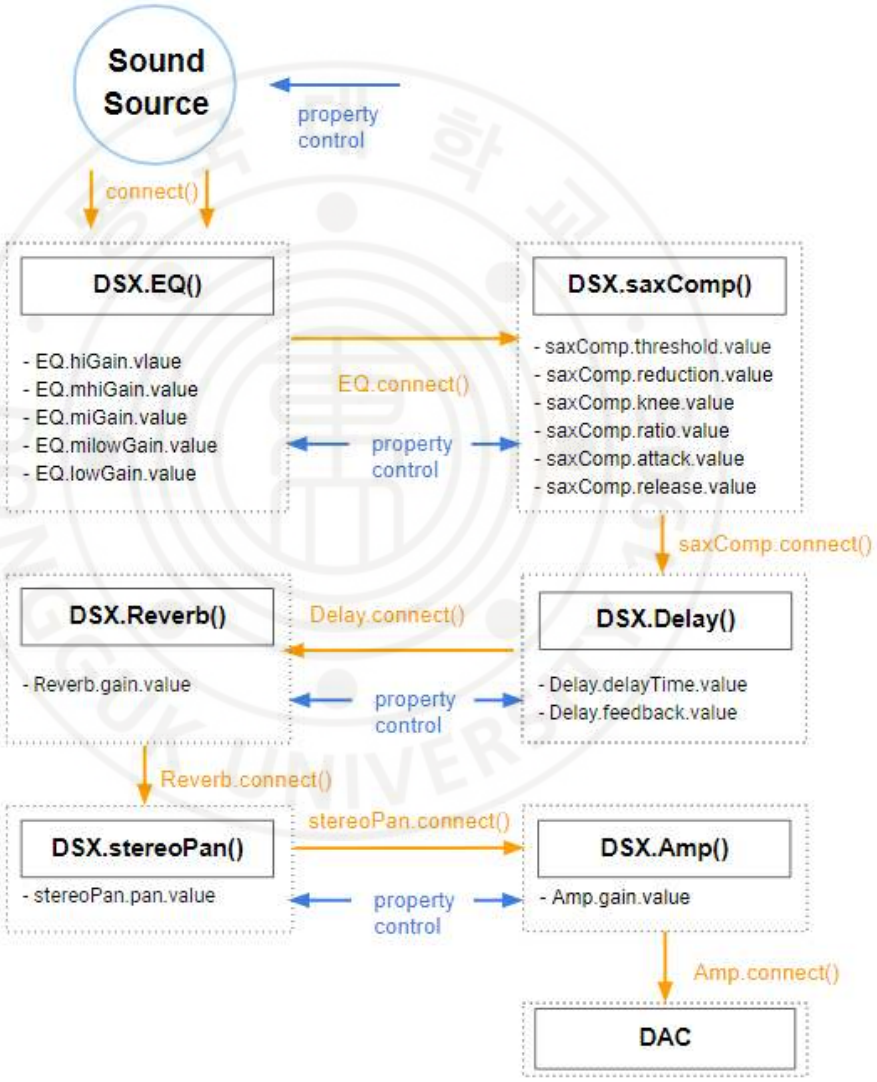
[그림-3.12] Sound Effect 객체 구성요소

[그림-3.12]는 Sound Effect 객체 종류와 각 객체의 속성 값과 메서드를 나타내고 있다. Sound Effect를 구성하는 인터페이스 객체의 종류와 기능은 <표-3.3>과 같다.

<표-3.3> Sound Effect 객체 종류와 기능

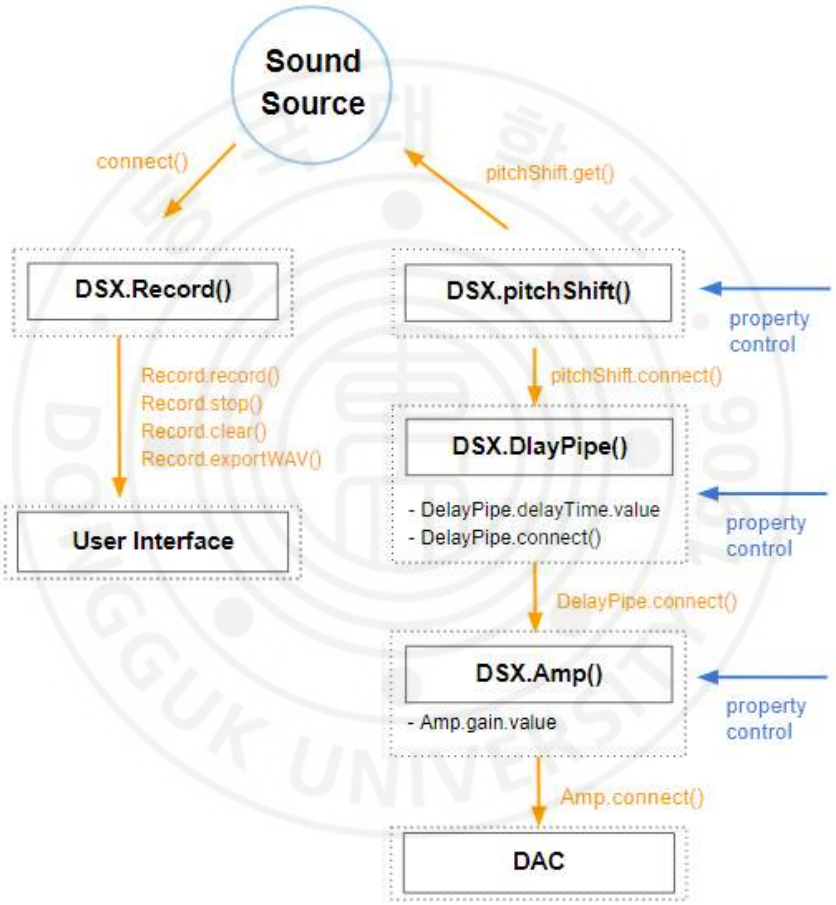
객체	기능
DSX.EQ()	사운드 소스를 5개의 주파수로 나누어 각 주파수의 음량을 제어한다.
DSX.saxComp()	사운드 소스의 음량과 다이나믹(dynamic)을 제어한다.
DSX.stereoPan()	사운드 소스를 left, right로 위치시키는 기능을 제공한다.
DSX.Delay()	사운드 소스를 지연시키고 피드백을 사용하여 지연되는 횟수를 제어한다.
DSX.Reverb()	사운드 소스에 잔향음과 반사음 효과를 제공한다.
DSX.Record()	사운드 소스를 녹음하여 오디오 파일을 제공한다.
DSX.pitchShift()	사운드 소스의 음색과 피치를 변조시킨다.
DSX.DelayPipe()	사운드 소스에 시간차를 두어 반복되는 loop 효과를 제공한다.

[그림-3.13]은 Sound Effect 객체 중 일반적으로 상용화되어 사용되는 이펙터 DSX.EQ(), DSX.saxComp(), DSX.Delay(), DSX.Reverb(), DSX.stereoPan()의 기본 라우팅 흐름도를 나타내고 있다.



[그림-3.13] Sound Effect 인터페이스 객체의 라우팅 흐름도

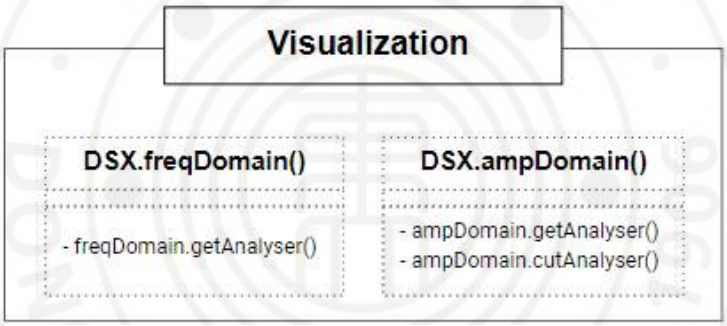
[그림-3.14]는 Sound Effect 객체 중 공연에 사용할 목적으로 개발된 사운드 프로세싱 이펙터 DSX.pitchShift()와 DSX.DelayPipe() 그리고 DSX.Record()의 기본 라우팅 흐름도이다.



[그림-3.14] 사운드 프로세싱 이펙터 객체의 라우팅 흐름도

2) Visualization

Visualization은 Sound Unit의 구성요소 중 출력 값을 제공하는 오실레이터와 신디사이저 등의 오디오 인터페이스 데이터를 입력받아 HTML5의 캔버스를 통해 비주얼라이제이션 하는 인터페이스를 제공한다. Visualization은 크게 2종류로 프리컨시 도메인과 앰플리튜드 도메인으로 구성되어 있다.



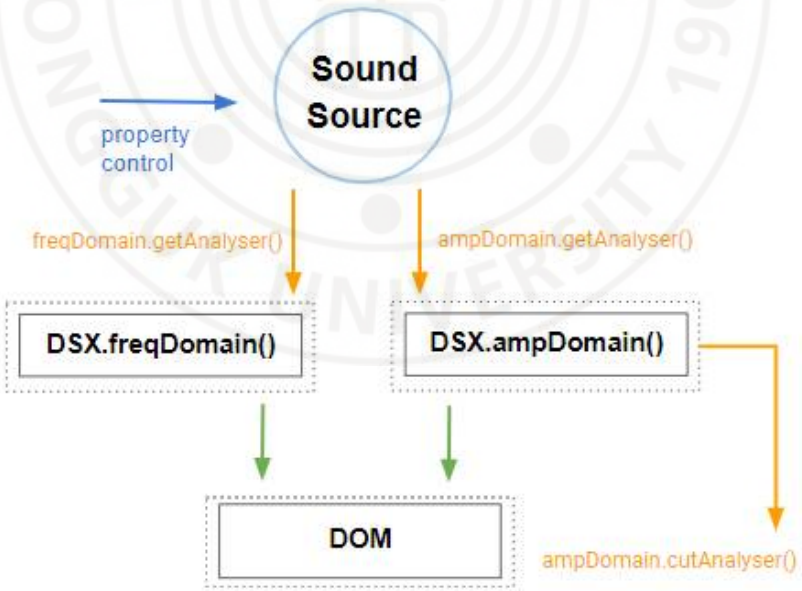
[그림-3.15] Visualization 객체 구성요소

Visualization은 기본적으로 사운드 출력 값을 모니터 하는 역할을 하고 있으며 프리컨시 도메인의 경우 이펙터가 효과적으로 적용되었는지 확인 가능하다. 또한 앰플리튜드 도메인은 음량이 over 되어 peak가 발생하는지 모니터 할 수 있다. [그림-3.15]는 Visualization 객체 종류와 각 객체의 메서드를 나타내고 있다.

<표-3.4> Visualization 객체 종류와 기능

객체	기능
DSX.freqDomain()	프리컨시 도메인으로 비주얼라이제이션
DSX.ampDomain()	엠플리튜드 도메인으로 비주얼라이제이션

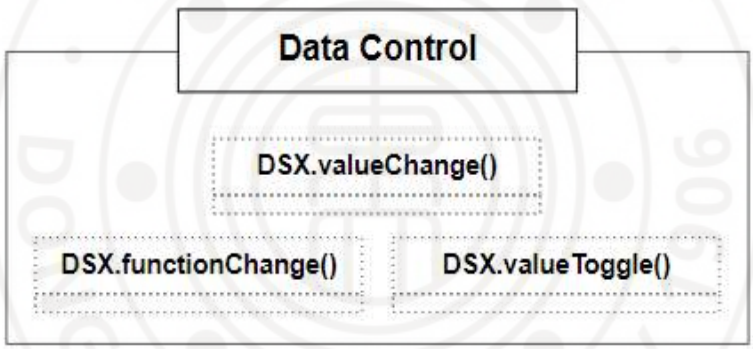
<표-3.4>는 Visualization 객체의 종류와 기능을 정의하고 있으며 [그림-3.16]은 사운드 소스의 출력 값을 입력받아 비주얼라이제이션 하는 라우팅 흐름도이다.



[그림-3.16] Visualization 라우팅 흐름도

3) Data Control

Data Control은 실시간으로 오디오 인터페이스의 음량 값이나 파라미터 값을 제어하고 오디오 인터페이스의 상태 값 변경을 위해 개발되었다. 3가지 타입의 인터페이스로 구성되어 있으며 간단한 신택스로 개발된 이벤트 객체 함수를 제공한다. [그림-3.17]은 Data Control의 객체 종류를 나타내고 있다.



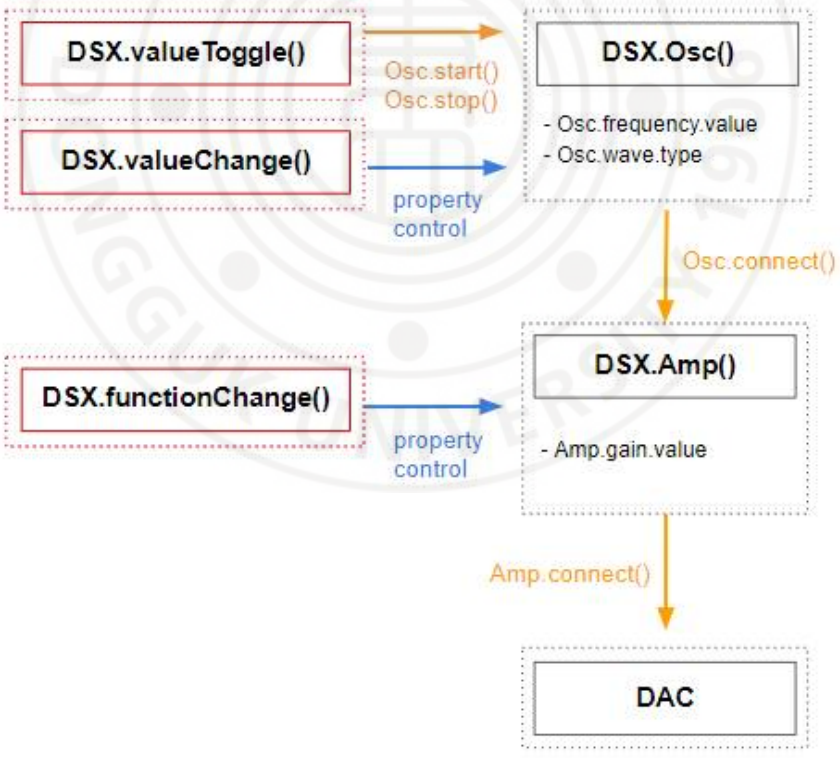
[그림-3.17] Data Control 객체 구성요소

Data Control 객체는 HTML의 input 태그나 button 태그 등의 id 속성 값과 연결되어 오디오 인터페이스의 파라미터 값과 on/off 설정을 제어한다. 또한 오디오 시스템 구현에 필요한 여러 가지 상태 값 제어 기능을 제공한다.

<표-3.5> Data Control 객체 종류와 기능

객체	기능
DSX.valueChange()	오디오 인터페이스의 파라미터 값을 제어하는 기능을 제공한다.
DSX.functionChange()	오디오 인터페이스의 상태 값을 함수로 제어하는 기능을 제공한다.
DSX.valueToggle()	on/off를 제어하는 토글 기능을 제공한다.

Data Control의 객체 종류와 기능은 <표-3.5>와 같으며 [그림-3.18]은 각 객체의 라우팅 흐름도를 나타내고 있다.



[그림-3.18] Data Control 라우팅 흐름도

2. DrSax.js 적용 방법 및 모델링

이번 절에서는 본 연구결과로 만들어진 DrSax.js 튜토리얼 사이트[3]에서 배포하는 DrSax.js를 사용하여 오디오 시스템 구현을 위한 환경설정과 여러 가지 구현 모델에 대해 알아보겠다. 또한 이 과정을 통해 어떻게 DrSax.js를 활용하여 다양한 웹 기반의 오디오 애플리케이션을 구현할 수 있는지 살펴본다.

```
<body>
  <!-- Drsax.js Download File -->
  <script src="./drsax.js"></script>
  <!-- Drsax.js CDN URL -->
  <script src="https://drwebsax.github.io/DrSax.js/js/drsax.js"></script>
  <script>

    // Declare an instance of DSX
    var DSX = new DSX;
    // Example Code...
    var osc = new DSX.Osc({type:"sawtooth", freq: 700});
    var gain = new DSX.Amp({ gain: 0.1});

  </script>
</body>
```

[그림-3.19] HTML의 DrSax.js 환경설정

3) https://drsax.github.io/DrSAX/lib.1.8.html.

먼저 DrSax.js를 기반으로 하는 애플리케이션을 구현하기 위해서는 DrSax.js 자바스크립트 파일을 HTML 페이지에 삽입하여 로딩 해야 한다. DrSax.js 파일은 튜토리얼 사이트에서 간단하게 다운로드 받거나 CDN(Content Delivery Network)[4] 주소를 사용할 수 있다.

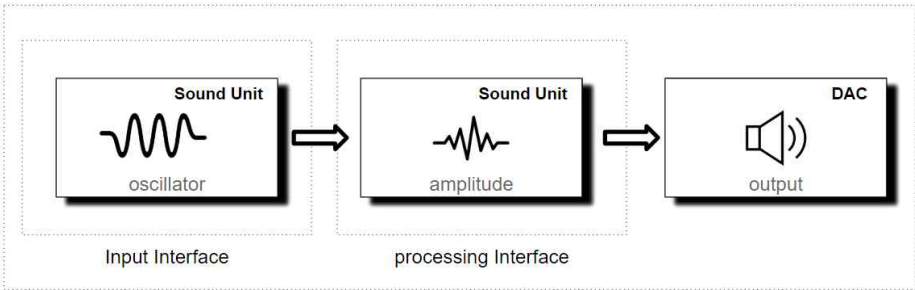
[그림-3.19]는 HTML에 DrSax.js의 초기 환경 설정을 설명하는 예제 코드이다. DrSax.js 파일을 HTML에 삽입 후 오디오 인터페이스 객체를 사용하기 위해 실행에 필요한 생성자 함수 DSX의 객체를 선언해야 한다. 선언된 DSX 객체 기반 위에 DrSax.js에서 제공하는 오디오 인터페이스 객체를 실행 하여 오디오 시스템을 구현할 수 있다.

다음은 오디오 시스템 구현을 위해 DrSax.js의 구성요소인 오디오 인터페이스 객체의 구현 모델에 대해 알아보겠다. 구현 모델은 Sound Unit의 구성요소인 Sound Synthesis, Sound Input, Audio Player 중 출력 값을 가지는 인터페이스를 중심으로 Visualization과 Data Control을 구현하는 방법에 대한 소스코드와 라우팅 프로세스를 제공한다. 먼저 Sound Unit의 Sound Synthesis 객체 DSX.Osc()의 오디오 시스템 구현 모델에 대해 알아보도록 하겠다.

DSX.Osc() 구현 모델의 프로세스 구조는 [그림-3.20]과 같으며 이를 바탕으로 구현된 모델은 [그림-3.21]과 같다. DrSax.js에서 제공하는 오디오 인터페이스의 객체는 선언과 동시에 속성 값을 설정할 수 있다.

---

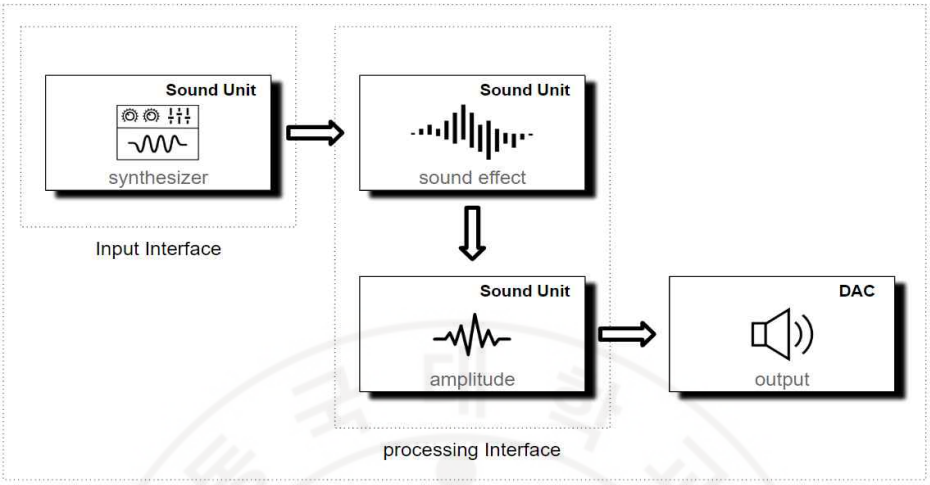
4) URL을 통해 동영상, 파일, 이미지 등의 다양한 콘텐츠를 안정적으로 제공하는 기술.



[그림-3.20] DSX.Osc() 구현 모델 프로세스 구조

```
// Declare an instance of DSX
var DSX = new DSX;

// Declare an Input Interface
var osc = new DSX.Osc({ type: "sawtooth", freq: 700 });

// Declare a Processing Interface
var gain = new DSX.Amp({ gain: 0.1 });

// Connection : osc to gain
osc.connect(gain);

// Connection : gain to Main Out
gain.connect(DAC);

// Start
osc.start();

// Stop
osc.stop();
```

[그림-3.21] DSX.Osc() 구현 모델 프로그래밍

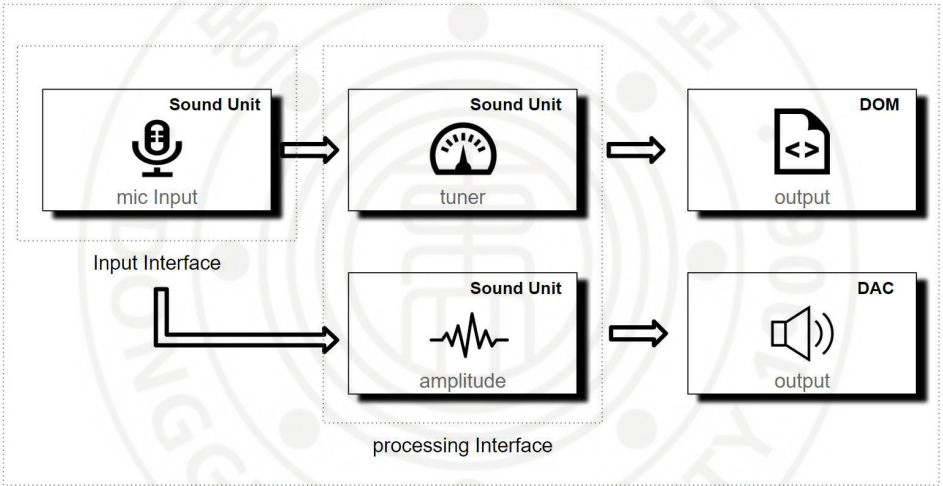


[그림-3.22] Sound Effect 구현 모델 프로세스 구조

```
// Declare an Input Interface
var fm = new DSX.FM({
  carrier_type: "sine", carrier: 500, mod_type : "sine",
  modfreq: 700, depth: 1800, gain: 0.5
});

// Declare a Sound Effect Interface
var Delay = new DSX.Delay({
  delayTime : 200, feedback: 0.45,
});

// Connection : fm to delay
fm.connect(delay);
```

[그림-3.23] Sound Effect 구현 모델 프로그래밍

[그림-3.22]는 Sound Unit의 Sound Effect 구현 모델의 프로세스 구조를 나타내고 있으며 [그림-3.23]은 Sound Effect 구현 모델 프로그래밍 코드를 나타내고 있다. 사운드 출력 인터페이스는 FM신디사이저로 설정하였고 딜레이 이펙터와 연결되어 오디오 시스템을 구현하였다. [그림-3.23]의 예제 코드를 기본으로 DrSax.js에서 제공하는 다양한 이펙터를 응용하여 오디오 시스템을 구현할 수 있다.



[그림-3.24] Sound Input 구현 모델 프로세스 구조

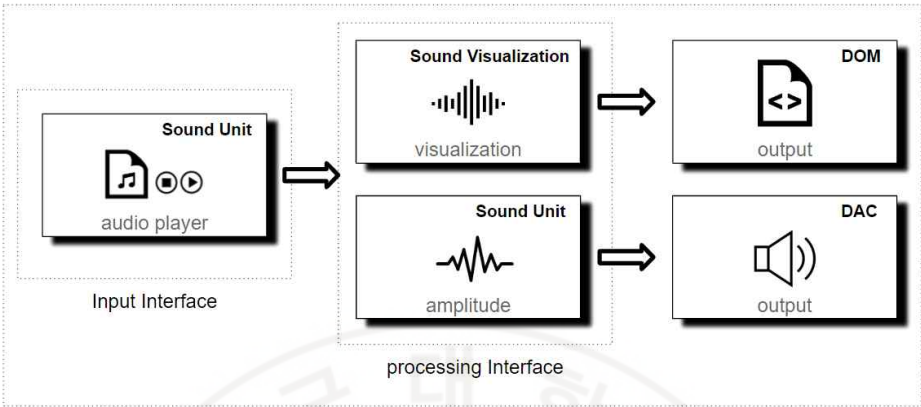
[그림-3.24]는 Sound Input 구현 모델의 프로세스 구조를 나타내고 있으며 [그림-3.25]는 Sound Input 구현 모델 프로그래밍 코드를 나타내고 있다. Sound Input의 DSX.Mic() 객체는 일반적으로 DAC로 연결되어 소리를 발생시키고 DSX.Tuner() 객체와 연결되어 시스템이 구현되며 튜너의 출력 값은 DOM으로 연결되어 피치 값을 보여준다.

```
// Declare an Amp Interface
var gain = new DSX.Amp({ gain: 0.5 });

// Declare a Mic Interface
var saxInput = new DSX.Mic();

// Declare a Tuner Interface
// Connection : tag id = 'pitch', 'note'
var tune = new DSX.Tuner({ 'pitch', 'note' });

// Connection : Mic Interface to gain
micInput.connect(gain);

// Connection : tune to micInput
tune.getAnalyser(micInput);
```

[그림-3.25] Sound Input 구현 모델 프로그래밍

[그림-3.26]은 DrSax.js의 구성요소인 Visualization의 적용 방법을 위한 구현 모델의 프로세스 구조를 나타내고 있으며 [그림-3.27]은 Visualization 구현 모델 프로그래밍 코드를 나타내고 있다. Audio Player의 객체 DSX.BGsound()는 DSX.Amp()와 연결되어 사운드를 재생하고 Visualization의 객체 DSX.freqDomain()은 DSX.BGsound()와 연결된 DSX.Amp()의 출력 값을 입력받아 DOM에 비주얼라이제이션 한다.

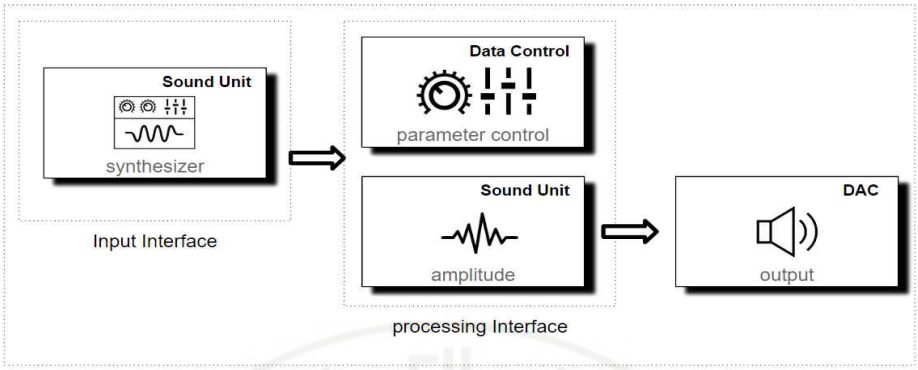
DSX.freqDomain() 객체는 HTML 캔버스 태그의 id와 연결하여 DOM에 비주얼라이제이션 할 수 있으며 색상 값을 설정할 수 있다.



[그림-3.26] Visualization 구현 모델 프로세스 구조

```
// Declare an Amp Interface
var gain = new DSX.Amp({ gain: 0.5 });

// Declare an Audio Plyer Interface
// Connection : input tag id = 'inputS'
var file_upload = new DSX.BGsound( 'inputS' );

// Declare a Visualization Interface
// Connection : Canvas tag id = 'canvs'
var freqcanvas = new DSX.freqDomain( 'canvs', 'red' );

// Connection : file_upload to gain
file_upload.connect(gain);

// Connection : freqcanvas to gain
freqcanvas.getAnalyser(micInput);
```

[그림-3.27] Visualization 구현 모델 프로그래밍



[그림-3.28] Data Control 구현 모델 프로세스 구조

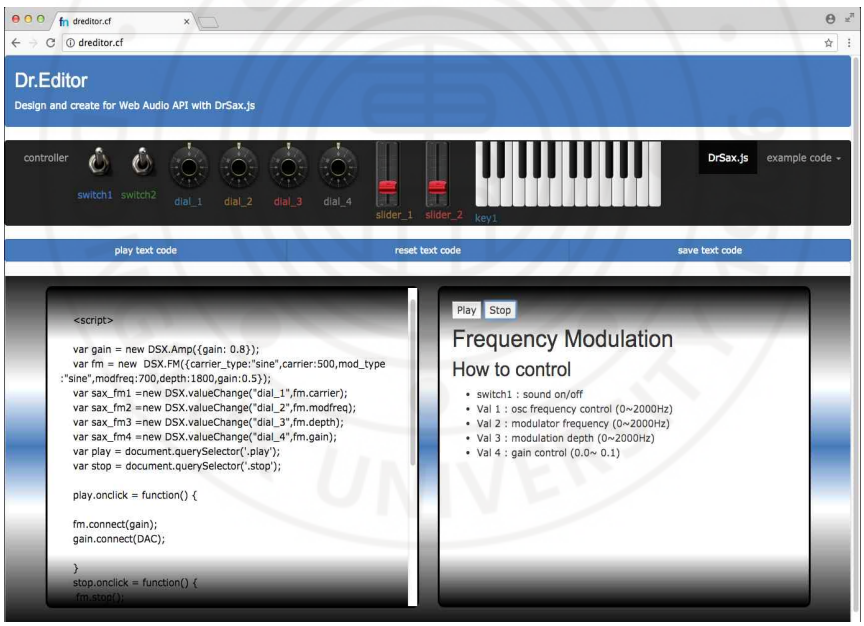
[그림-3.28]은 DrSax.js의 구성요소인 Data Control 구현 모델의 프로세스 구조를 나타내고 있으며 [그림-3.29]는 Data Control의 구현 모델 프로그래밍 코드를 나타내고 있다. [그림-3.29]의 출력 인터페이스는 FM 신디사이저로 설정하였고 DSX.Amp() 객체와 연결되어 오디오 시스템을 구현하였다. 또한 데이터 컨트롤의 3가지 중 DSX.valueChange() 객체를 적용하여 FM 신디사이저의 파라미터 값을 제어하는 프로그래밍을 보여주고 있다. [그림-3.29]의 예제 코드를 기본으로 Data Control의 3가지 객체를 응용하여 오디오 인터페이스 객체의 파라미터 값 및 이벤트 함수의 상태 값 제어가 가능하다.

```
// Declare an Input Interface
var fm = new DSX.FM({
  carrier_type: "sine", carrier: 500, mod_type : "sine",
  modfreq: 700, depth: 1800, gain: 0.5
});

// Declare an Amp Interface
var gain = new DSX.Amp({ gain: 0.5 });

// Declare a Data Control Interface
// Connection : tag id = 'knb'
var fm_value = new DSX.valueChange( "knb", fm.carrier );

// Connection : osc to Main out
fm.connect(gain);
gain.connect(DAC);
```

[그림-3.29] Data Control 구현 모델 프로그래밍

3. 라이브러리 활용을 위한 웹 에디터 구현

DrSax.js 기반의 오디오 시스템 구현은 다양한 프로그래밍 작업과 디버깅 테스트를 필요로 한다. 현재 웹 브라우저는 개발자 도구[5]를 기본으로 제공하고 있지만 DrSax.js로 구현된 코드를 디버깅하고 UI 컨트롤 테스트와 복잡한 프로그래밍을 구현하기에는 한계가 있다. 이러한 이슈를 해결하고 빠르고 쉬운 웹 오디오 애플리케이션 개발을 위해 웹 기반의 DrEditor를 개발하였다.



[그림-3.30] DrEditor UI 디자인

5) 브라우저에서 제공하는 편집 창으로 개발에 필요한 코드 에디팅, 디버깅 등의 기능 및 도구를 제공하는 개발 환경.

DrEditor는 오디오 시스템 구현에 필요한 다음과 같은 기능을 제공한다.

- 버튼, 다이얼, 피아노 건반 등의 UI 컨트롤러
- DrSax.js 기반으로 구현된 <표-3.6>의 프리셋을 제공
- 편집된 소스코드 텍스트 파일로 다운로드
- 웹 기반의 접근성

또한 DrSax.js 라이브러리 이외에도 Web Audio API와 HTML5, CSS, 자바스크립트 사용이 가능해 일반 에디터 기능과 교육용 애플리케이션으로 활용이 가능하다.

<표-3.6> DrEditor 프리셋 종류

	프리셋 종류
Synthesizer	- simple osc - frequency modulation - amplitude modulation
Sound Effect	- reverb - delay - panning
Audio Player	- sound player

Ⅳ. 인터랙티브 웹 애플리케이션 설계 및 구현

1. 웹 오디오 시스템을 위한 기술적용

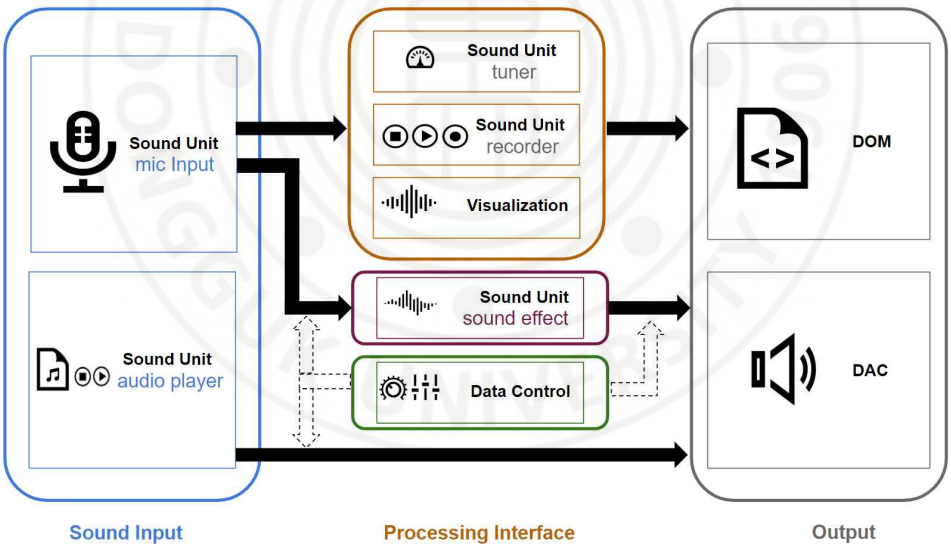
본 절에서는 DrSax.js와 웹 기술을 적용하여 인터랙티브 공연에 사용 가능한 인터랙티브 웹 애플리케이션 개발을 목적으로 하고 있다. 실시간 인터랙션을 통해 사운드 프로세싱과 이펙터 제어가 가능하고 연주에 필요한 여러 가지 기능을 제공한다. 인터랙티브 멀티미디어 공연을 위해 제작된 웹 애플리케이션 DrWebSax에 대해 살펴보도록 하겠다.

1) 오디오 시스템 아키텍처

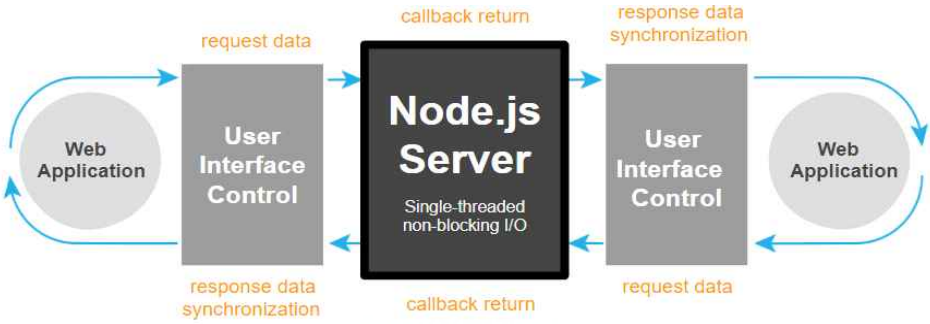
DrWebSax는 기본적으로 DrSax.js의 인터페이스를 적용하여 연주에 필요한 사운드 컴포넌트로 구현하였으며 UI는 [그림-4.1]과 같다.

[그림-4.1] DrWebSax 애플리케이션의 모바일 UI

사운드 컴포넌트는 마이크 인풋과 사운드 프로세싱, 사운드 이펙터 등으로 구성되어 있고 공연에 필요한 다양한 유틸리티 컴포넌트도 제공하고 있다. [그림-4.2]는 DrWebSax의 사운드 프로세스 구조를 나타내고 있다. 마이크를 통한 사운드 인풋 값은 Tuner 컴포넌트에 전달되어 피치 값을 출력하고 DrWebSax의 사운드 분석을 통해 프리컨시 도메인으로 비주얼라이제이션 된다. 또한 Recorder 컴포넌트를 통해 녹음된 오디오 파일을 제공하며 Audio Player를 사용하여 연주에 필요한 오디오 재생도 가능하다. DrWebSax의 이펙터 컴포넌트는 입력받은 사운드를 프로세싱하며 데이터 컨트롤 컴포넌트를 통해 실시간으로 제어된다.

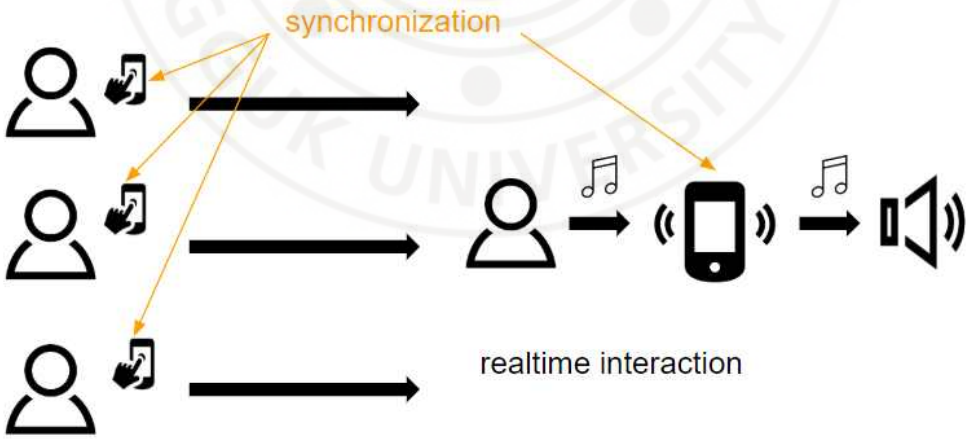


[그림-4.2] DrWebSax 프로세스 구조



[그림-4.3] DrWebSax의 네트워킹 시스템 흐름도

DrWebSax의 실시간 인터랙티브 시스템은 Node.js의 Socket.io 모듈을 사용하여 구현하였다. Socket.io는 PC와 모바일간의 실시간 네트워킹 기술을 제공하여 사운드를 제어하는데 매우 효과적이며 활용도가 높다. 또한 DrWebSax에서 제어되는 사운드 데이터와 UI는 실시간 인터랙션을 통해 동기화되어 직관적인 모니터링이 가능하다.

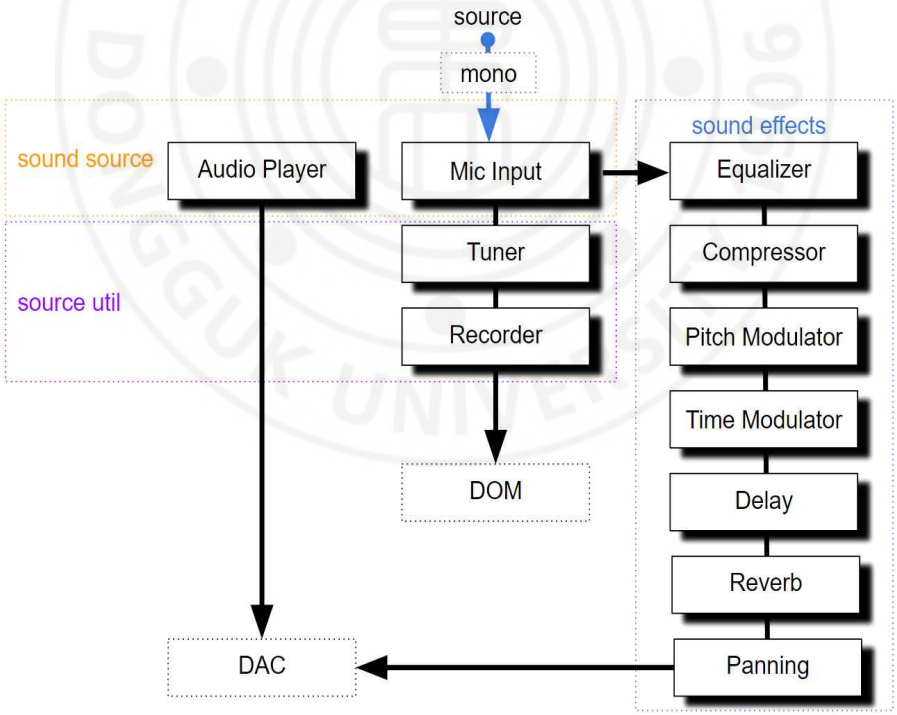


[그림-4.4] 네트워킹을 통한 실시간 오디오 시스템 제어

2) 오디오 시스템 컴포넌트 구성

DrWebSax는 공연에 필요한 11개의 오디오 컴포넌트로 구성되며 기능에 따라 다음과 같이 구분하였다.

- 사운드 소스를 제공하는 컴포넌트
- 사운드 소스의 출력 값을 활용하여 데이터 값을 UI에 제공하는 컴포넌트
- 사운드 이펙터 컴포넌트



[그림-4.5] 오디오 시스템 컴포넌트의 종류와 라우팅 흐름도

사운드소스를 제공하는 컴포넌트는 Audio Player와 마이크 인풋으로 구분된다.

먼저 Audio Player는 공연에 필요한 오디오 파일을 재생하기 위한 목적으로 구성되었다. 재생 가능한 오디오 파일은 브라우저와 모바일 기종에 따라 <표-4.1>과 같다 [58].

<표-4.1> 브라우저와 모바일 기종에 따른 재생 가능한 파일

브라우저	오디오 파일 형식
크롬	AAC, MP3, OGG, WAV
인터넷 익스플로러	AAC, MP3
iOS, 아이패드	AAC, AIF, MP3, WAV
안드로이드	AAC, M4A, MID, MP3, OGG, WAV

* 2019년 3월16일 기준 [58]

사운드소스를 제공하는 또 다른 컴포넌트인 Mic Input 컴포넌트는 DrWebSax의 마이크 값을 제어하는 기능을 제공하며 Tuner 컴포넌트와 Recorder 컴포넌트 그리고 사운드 이펙터 컴포넌트로 연결되어 활용된다.

Tuner 컴포넌트는 마이크 인풋 값의 정확한 음정을 체크를 위해 개발하였다. 모든 관악기는 온도나 주변 환경에 의해서 음정이 수시로 변하게 되어 튜너가 필수적으로 있어야 한다. Tuner 컴포넌트는

Mic Input 컴포넌트와 연결되어 피치 값을 출력한다.

Recorder 컴포넌트는 사운드 녹음 및 재생을 위해 개발되었으며, 다운로드가 가능한 녹음 파일을 제공한다. 녹음된 파일은 오디오 에디팅을 통해 음원소스로 사용할 수 있다.

앞서 소개된 Tuner 컴포넌트와 Recorder 컴포넌트는 마이크 인풋 값을 활용하여 결과 값을 UI 화면에 제공한다. 마이크 인풋 값을 사용하는 또 다른 사운드 이펙터 컴포넌트는 사운드를 변화시켜 공연에 중요한 기능을 제공한다. 본 DrWebSax에 사용된 11개의 사운드 이펙터에 대해 살펴보겠다.

● **Equalizer 컴포넌트**

Equalizer 컴포넌트는 사운드 인풋 값의 주파수별 음량을 제어하기 위해 개발되었으며 5개의 주파수 값을 슬라이더로 제어할 수 있다. 6개의 프리셋을 설정하여 장르별로 EQ 세팅을 사용할 수 있다. 이퀄라이저 컴포넌트 프리셋의 종류는 voice, pop, dance, rock, metal, classic이다.

● **Compressor 컴포넌트**

Compressor 컴포넌트는 사운드 인풋 값의 다이나믹과 음량을 제어할 수 있는 기능을 제공한다. 6개의 속성 값을 제어할 수 있고 5

개의 프리셋을 설정할 수 있다. 프리셋의 종류는 vocal, brass, guitar, drum, dance 이다.

● Pitch Modulator 컴포넌트

Pitch Modulator 컴포넌트는 음정과 음색을 변화시키는 효과를 위해 개발하였으며, 다양한 사운드 프로세싱이 가능하다. Pitch Modulator 컴포넌트와 Delay 컴포넌트를 같이 사용하면 효과적으로 사운드를 극대화 시킬 수 있다.

● Time Modulator 컴포넌트

Time Modulator 컴포넌트는 사운드의 시간차를 이용해 루핑(looping) 효과를 사용하기 위해 개발되었다. 4개의 딜레이를 활용하여 시간차를 설정할 수 있고 시간차 범위는 각각 0~1초, 0~2초, 0~3초, 0~4초이다.

● Delay/Reverb 컴포넌트

Delay/Reverb 컴포넌트는 공간계 이펙터로, 사운드에 딜레이 효과와 반향, 잔향을 사용하여 사운드를 풍부하게 해준다. 또한 딜레이의 딜레이타임 값과 피드백 값은 효과적인 사운드 프로세싱으로도 사용 가능하다. 프리셋의 종류는 zero, space, vocal, solo 이다.

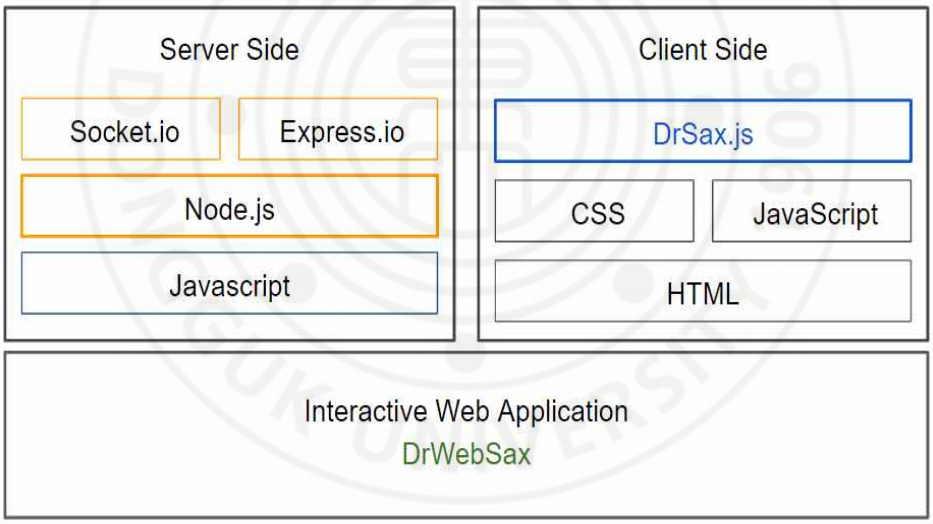
● Panning 컴포넌트

Panning 컴포넌트는 사운드 소스의 left, right 위치 변화를 위해 개발하였다. 슬라이더를 통해 값을 제어할 수 있으며 left, right 값의 범위는 0~100으로 설정할 수 있다.

3) 프로그래밍 및 오디오 시스템 구현

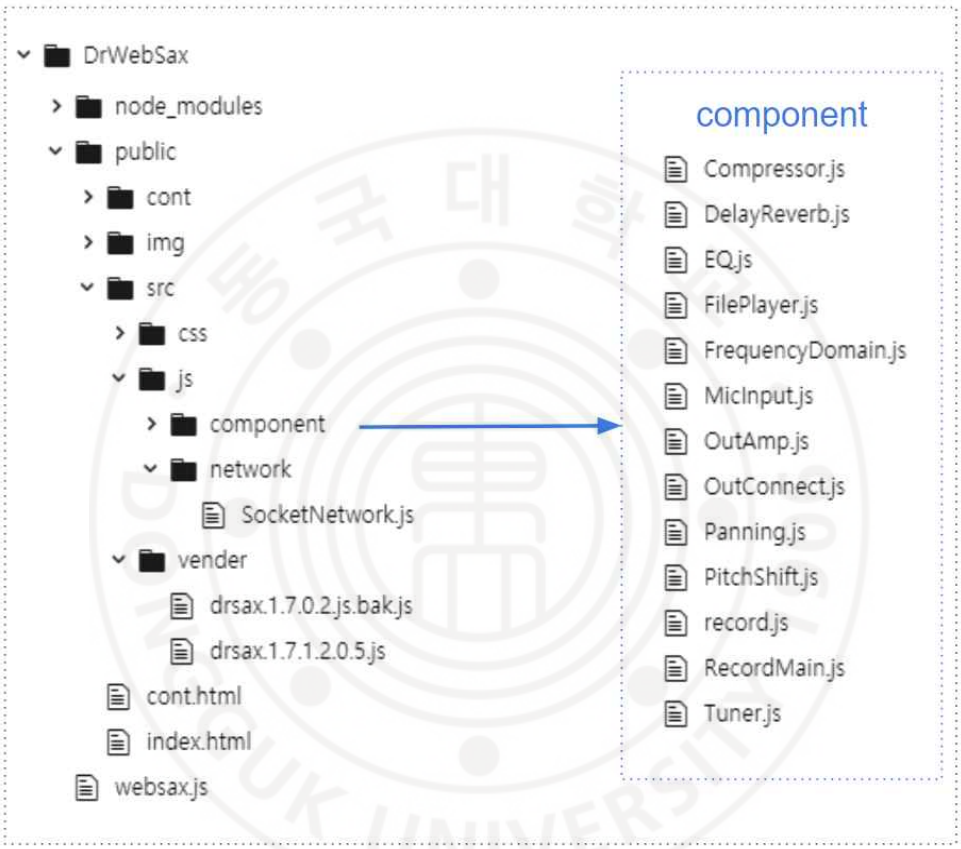
DrWebSax는 오디오 시스템 구현을 위해 앞에서 설계된 컴포넌트를 중심으로 오디오 시스템을 구현하였고 실시간 인터랙션 시스템을 위해 Node.js를 사용하였다.

전체적인 프로그래밍 구조는 [그림-4.6]과 같다. 클라이언트 영역은 HTML5와 CSS, 자바스크립트기반으로 DrSax.js를 사용하였으며 서버사이드 영역은 Node.js를 기반으로 서버를 구축하였다.



[그림-4.6] DrWebSax 프로그래밍 구조

[그림-4.7]은 DrWebSax의 프로그래밍 폴더 구조와 컴포넌트의 js 파일 구성을 보여주고 있다.



[그림-4.7] DrWebSax의 디렉토리 구조

(1) 오디오 시스템 프로그래밍

DrWebSax는 11개의 컴포넌트로 프로그래밍되어 있으며 각각의 컴포넌트의 특징과 기능에 따라 유기적으로 연결되어 있다. [그림-4.8]은 DrWebSax의 DrSax.js 초기 환경설정과 DSX의 객체 선언, 실시간 인터랙션을 위한 Socket.io 설정 그리고 삽입된 DrSax.js와 컴포넌트 파일을 보여주고 있다.

```
<script src="https://drwebsax.github.io/DrSax.js/js/drsax..js"></script>
<script src="/socket.io/socket.io.js"></script>
<script>
  var DSX = new DSX;
  var socket = io();
</script>
<script src="./src/js/component/OutAmp.js"></script>
<script src="./src/js/component/Syn.js"></script>
<script src="./src/js/component/MicInput.js"></script>
<script src="./src/js/component/RecordMain.js"></script>
<script src="./src/js/component/EQ.js"></script>
<script src="./src/js/component/Compressor.js"></script>
<script src="./src/js/component/DelayReverb.js"></script>
<script src="./src/js/component/Panning.js"></script>
<script src="./src/js/component/OutConnect.js"></script>
<script src="./src/js/component/FrequencyDomain.js"></script>
<script src="./src/js/component/Tuner.js"></script>
<script src="./src/js/component/FilePlayer.js"></script>
<script src="./src/js/component/PitchShift.js"></script>
<script src="./src/js/network/SocketNetwork.js"></script>
```

[그림-4.8] DrWebSax의 HTML 마크업

DrWebSax의 컴포넌트 구조는 앞서 연구된 3장 2절의 Sound Input 구현 모델을 기반으로 프로그래밍 되었으며 [그림-4.9]는 사운드 입력 값을 제어하기 위해 DSX.Mic() 객체를 사용한 소스코드를 나타내고 있다.

```
// mic input
var saxInput = new DSX.Mic();

// sax input on/off control
var saxOnoff = new DSX.functionChange("saxmic_on", sax_on_data);

function sax_on_data(e) {
  if (MICINPUT_ONOFF === true) {
    saxInput.connect(saxgain);
    if (e.target.value === "1") {
      saxgain.connect(gain);
      saxgain.gain.value = 0.7;
      tune.getAnalyser(saxgain);
      sax_canvas_freq.getAnalyser(saxgain);
    }
    if (e.target.value === "0") {
      saxgain.disconnect(0);
    }
  }
}
```

[그림-4.9] DrWebSax의 DSX.Mic() 프로그래밍

[그림-4.10]은 DrWebSax에 구성된 컴포넌트 라우팅을 보여주고 있다. 라우팅 연결을 위해 사용된 DSX.Amp()는 사운드 믹싱과 사운드 음량을 제어한다.

```
saxGain.connect(gain);                  // saxGain -> Mic Input
gain.connect(gainEq);                   // gainEq -> EQ
gainEq.connect(gainComp);               // gainComp -> Compressor
gainComp.connect(gain4);                // Aux1
gain4.connect(gainPitchMod);            // gainPitchMod -> Pitch Modulator
gainPitchMod.connect(gain6);            // Aux2
gain6.connect(gainDelay);               // gainDelay -> Delay
gainDelay.connect(gainReverb);          // gainReverb-> Reverb
gainReverb.connect(gainTimeMod);        // gainTimeMod -> Time Modulator
gainTimeMod.connect(gain10);            // Aux3
gain10.connect(mixgain);                // Aux4
gain.connect(saxInputSub);              // Mic Input Aux
saxInputSub.connect(mixgain);           // Aux5
mixgain.connect(saxPan);                // saxPan -> Panning
saxPan.connect(mainOut);                // main out
mainOut.connect(DAC);                   // main out -> DAC
```

[그림-4.10] DrWebSax의 믹싱 라우팅

(2) 오디오 시스템의 인터랙티브 네트워킹

본 연구에서는 실시간 인터랙티브 네트워킹 통신을 위해 Node.js의 Socket.io 모듈을 사용하였다. Socket.io는 비동기방식으로 데이터를 처리하는 기능을 제공해 실시간 인터랙티브 네트워킹 시스템에 사용되었으며 [그림-4.11]은 Node 서버의 Socket.io 프로그래밍을 나타내고 있다.

```
// socket connect
io.on('connection', function(socket){
  // Mic out
  socket.on('micOut', function(data){
      io.emit('micOut',data);
  });
  // Audio file
  socket.on('audioFile', function(data){
      io.emit('audioFile',data);
  });
```

[그림-4.11] Node 서버의 네트워킹 설정

실시간 인터랙션을 위한 네트워크 통신 시스템은 [그림-4.12]와 같은 구조로 구성되어 있다. DrWebSax 콘트롤러의 (a) 슬라이더로 마이크 인풋 값을 제어하면 (b) socket의 콜백 함수 인자 data를 통해 (c)의 io.emit data 인자로 전달된다.

[그림-4.12] 마이크 인풋 속성 값의 서버 네트워킹 통신

[그림-4.12]의 (c) data 인자 값은 (d) 콘솔로그를 통해 (d-1)에 출력된다. (d-1)의 saxmic_gain 값은 0.21을 나타내고 있으며 백분위로 환산되어 (d)에 메인 DrWebSax 마이크 인풋 슬라이더에 21%로 전달되어 마이크 입력 값을 제어한다.

2. 미디어 설치작품에서의 기술적용

본 시스템은 DrSax.js와 웹 기술을 활용한 미디어 설치 작품 구현을 목적으로 한다. 미디어아트의 특성을 살려 관객이 작품에 직접 참여하고 웹 브라우저와 접근성에 있어 장점을 가진 모바일을 사용하여 작품에 접근할 수 있는 방법을 제공한다. 앞서 연구된 기술을 적용한 인터랙티브 미디어 설치작품 Composition 2017[1]에 대해 살펴보도록 하겠다.

1) 설치작품 아키텍처

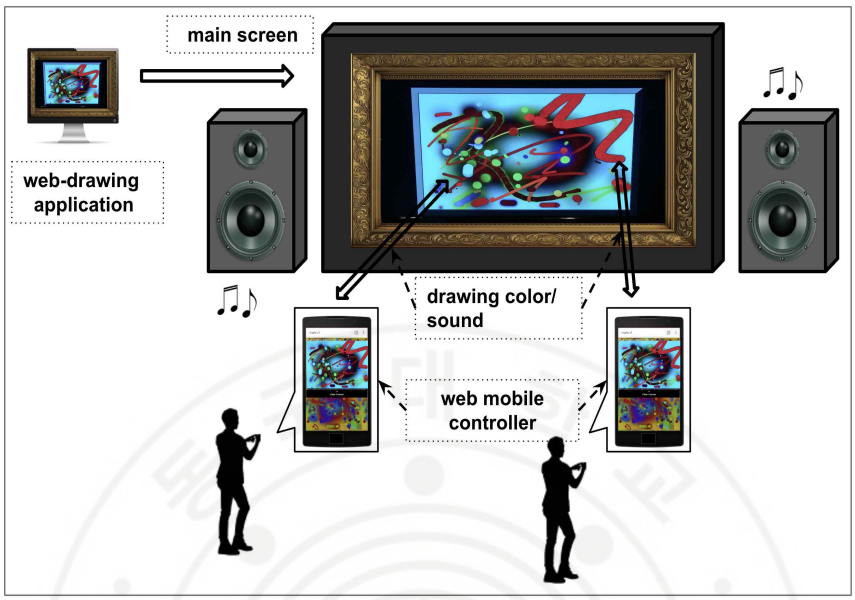
본 연구는 칸딘스키(Kandinsky)[2] 작품의 추상적인 이미지 색상과 형태에 영감을 받아 구현된 미디어 설치 작품으로 실시간 인터랙션을 통해 즉흥적인 이미지 제어와 음악적 표현을 목적으로 하고 있다. 이러한 작품 구현을 위해 관객이 작품에 직접 참여하여 실시간으로 이미지를 그리고 이를 통해 사운드를 제공하는 시스템을 설계하였다. 작품에 참여한 여러 관객이 모바일을 사용하여 그림을 그리면 메인 화면에 실시간으로 그림이 그려진다. 즉흥적인 표현으로 그려지는 이미지의 색상, 넓이 등의 데이터는 메인화면의 이미지와 사운드를 제어하는데 사용된다. 추상적인 사운드 표현을 위해 DrSax.js의 FM 신디사이저를 사용하였다.

[그림-4.13]은 설치작품의 시스템구조를 나타내고 있다.

---

1) 본 연구자가 구현한 미디어 설치작품으로 2017년 갤러리 31 미술관에서 개최된 “Seeing Sound Listening Image” 전시회에 출품되었다. https://vimeo.com/223936815.

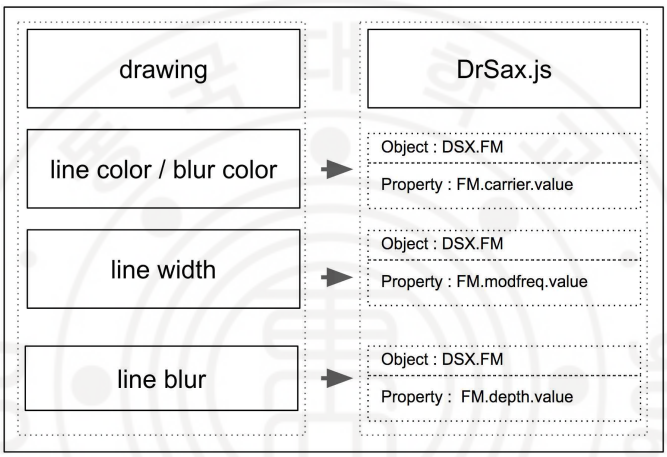
2) 1866 ~ 1944, 러시아 출생의 추상주의 화가, 공감각 아티스트.



[그림-4.13] 설치작품 구조 흐름도

본 작품의 추상적인 이미지는 붓으로 그림을 그리듯 모바일 화면에 손가락을 터치하여 표현된다. 모바일의 이미지는 넓이(width), 블러(blur), 색상(color) 데이터에 따라 서로 다른 추상적 색상과 형태를 표현한다. 이러한 이미지를 모바일에서 제어하기 위해 HTML5의 캔버스 API를 사용하였으며 자바스크립트의 터치 함수와 캔버스의 드로잉 함수를 사용하였다. 이미지 색상의 주파수는 공감각을 통해 사운드의 주파수로 표현될 수 있다. 본 작품에서는 이미지와 사운드의 공감각적 인터랙션을 위해 본 작품에 사용된 색상의 주파수 값과 FM 신디사이저의 주파수 값을 대응시켰다. 그려지는 이미지 형태를 표현하는 선의 넓이와 블러 넓이의 데이터 값은 FM 신디사이저의 속성 값에 적용되어 사운드를 프로세싱

한다. 선의 넓이는 모듈레이션 주파수 값을 제어하고 블러 넓이는 뎁스(depth) 값을 제어한다. [그림-4.14]는 이미지 선과 블러 색상, 선의 넓이, 블러 넓이에 의해 실시간 제어되는 FM 신디사이저의 속성 값 설정을 보여주고 있다.



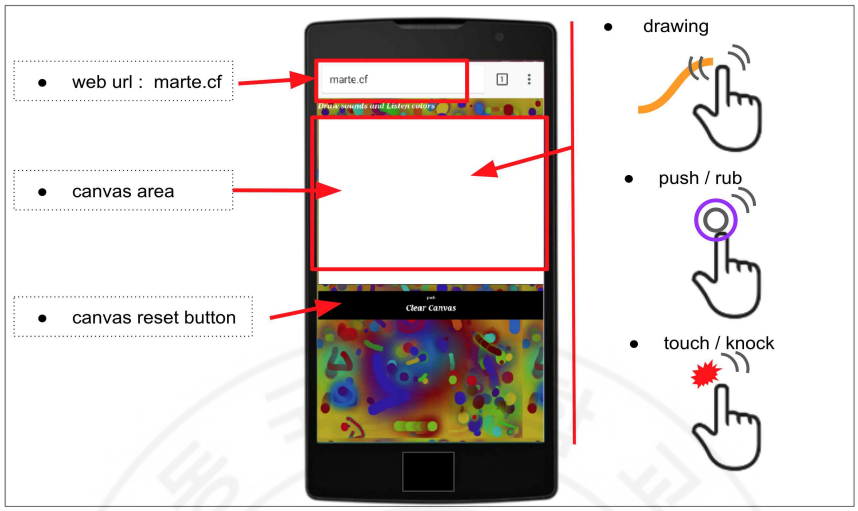
[그림-4.14] 이미지와 FM 신디사이저 사운드설정

이미지와 사운드를 제어하기 위해 모바일 컨트롤러의 UI를 두 가지 영역으로 구분하였다. 첫 번째는 그림을 그리는 캔버스 영역이고 두 번째는 그림을 지우는 리셋 버튼이다. 모바일의 캔버스에 그려진 이미지 데이터 정보는 실시간으로 메인 화면의 이미지와 동기화되고 사운드 데이터 값을 변화시킨다. 동기화된 메인 화면의 이미지는 모바일의 리셋버튼을 통해 그려진 화면을 지우고 새로 설정된 배경화면으로 리셋된다. 리셋 설정에 의해 배경 화면은 공감각 기반으로 설정된 색상을 사용하여

랜덤하게 그려진다. [그림-4.15]는 모바일을 통해 실시간으로 메인 화면에 그려진 이미지색상과 형태를 나타내고 있다.

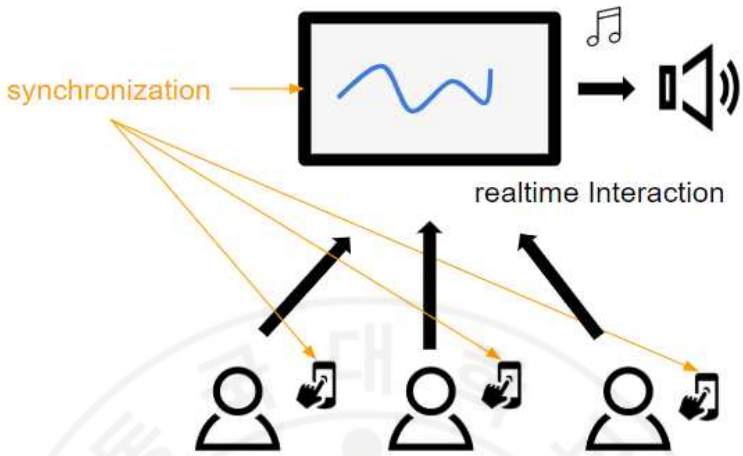
[그림-4.15] 메인 화면 애플리케이션

[그림-4.15]의 이미지는 손가락의 다양한 터치 방법에 따라 점, 선, 원 등의 형태로 그려진다. 예를 들어 피아노 건반을 연주하는 것처럼 두드리거나 한 점을 계속 문지르는 기법 등을 사용하여 다양한 이미지를 표현할 수 있으며 이러한 이미지 표현 방법은 색상과 블러 넓이 등을 랜덤하게 변화시킨다. 이런 표현방법에 따른 이미지 데이터 값은 실시간으로 사운드의 파라미터 값에 적용되어 사운드를 제어한다. [그림-4.16]은 모바일 컨트롤러의 기능과 다양한 터치 표현방법을 나타내고 있다.



[그림-4.16] 모바일 컨트롤 애플리케이션의 기능 및 특징

본 설치작품은 실시간 인터랙티브 시스템을 구축하기 위해 비동기 방식인 Node.js의 Socket.io 모듈을 사용하여 네트워킹 서버를 구축하였다. Node.js의 비동기 방식은 응답속도의 지연이 거의 없어 본 설치작품의 실시간 네트워킹 통신에 매우 효과적으로 사용되었다. Wi-Fi나 3G, 4G 데이터를 통해서 네트워킹 시스템에 접근 할 수 있으며 여러 관객들의 동시 참여가 가능하다. 관객은 접근성이 좋은 모바일의 웹 브라우저를 통해 실시간으로 본 작품의 인터랙션을 제어한다. [그림-4.17]은 네트워킹을 통해 실시간으로 설치작품을 제어하는 흐름도를 나타내고 있다.



[그림-4.17] 네트워킹을 통한 실시간 설치작품 제어

2) 프로그래밍 및 설치작품 시스템 구현

(1) 이미지와 사운드 제어

Composition 2017은 이미지가 그려질 때마다 12개의 색상 값과 주파수 값을 랜덤하게 출력하는 API를 호출한다.

```
var RandomValue =Math.floor(Math.random() *13),
  ColorList=["#57009f","#740000","#b30000","#ed0000","#ff6300",
  "#ffeb00","#9dff00","#29ff00","#00ffe3","#007dff","#4700e4","#4400eb"],
  PitchList=[349,370,392,415,440,466,493,523,554,587,622,659];
var RandomPitchSetValue =Math.floor(Math.random() *4);
var ExtendPitchData;
switch(RandomPitchSetValue) {
  case 0:
    ExtendPitchData=PitchList[RandomValue] / 2; break;
  case 1:
    ExtendPitchData=PitchList[RandomValue] * 1; break;
  case 2:
    ExtendPitchData=PitchList[RandomValue] * 2; break;
  case 3:
    ExtendPitchData=PitchList[RandomValue] * 3; break;
}
var RandomColorBlur =Math.floor(Math.random() *13);
var RandomWidth =Math.floor(Math.random() * 45) + 5;
var RandomBlur =Math.floor(Math.random() * 200);
return (
  [ColorList[RandomValue],ExtendPitchData,
  RandomValue,RandomWidth,RandomBlur,ColorList[RandomColorBlur]]
)
```

[그림-4.18] 이미지 설정 API

```
var ReturnData= DRAW.Framework.RandomSet();
RANDOMCOLOR= ReturnData[0];
RANDOMPITCH= ReturnData[1];
RANDOMDATA= ReturnData[2];
RANDOMWIDTH= ReturnData[3];
RANDOMBLUR= ReturnData[4];
RANDOMCOLORBLUR= ReturnData[5];
```

[그림-4.19] API 리턴 데이터의 활용

랜덤한 색상 값과 주파수 값을 출력하는 API 코드는 [그림-4.18]과 같으며 6개의 리턴 데이터를 출력한다. 6개의 출력 값은 [그림-4.19]와 같이 각각의 변수로 저장되어 선의 넓이, 주파수 색상, 블러 등의 이미지를 제어한다. <표-4.2>은 API 리턴 데이터 설정을 보여주고 있다.

<표-4.2> API 리턴 데이터 설정

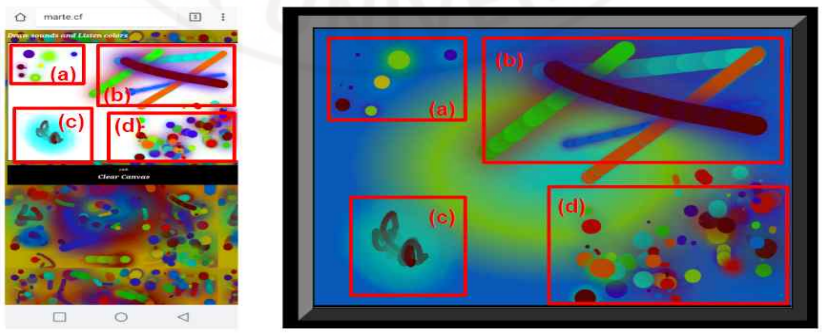
	랜덤 설정 값
컬러	"#57009f","#740000","#b30000","#ed0000", "#ff6300","#ffeb00","#9dff00","#29ff00", "#00ffe3","#007dff","#4700e4","#4400eb"
피치(Hz)	349, 370, 392, 415, 440, 466, 493, 523, 554, 587, 622, 659
옥타브	0.5, 1, 2, 3
색상블러	0~13
넓이	0~50
블러	0~200

모바일에 이미지를 그리기 시작하면 실시간 인터랙션을 통해 [그림-4.20]의 메서드가 실행되어 메인 화면에 이미지가 그려진다. 이를 통해 모바일과 메인화면의 이미지는 동기화된다.

```
context.lineWidth = width;
context.strokeStyle = color;
context.lineCap ="round";
context.shadowBlur = blurWidth;
context.shadowColor = blurColor;
context.beginPath();
context.moveTo(oldPoint.x, oldPoint.y);
context.lineTo(newPoint.x, newPoint.y);
context.stroke();
```

[그림-4.20] 캔버스 그림 설정

[그림-4.21]은 모바일과 메인화면의 동기화된 모습이며 터치방법에 따른 다양한 이미지이며 점찍기(a), 선 긋기(b), 문지르기(c), 두드리기(d)를 사용한 캔버스 화면을 나타내고 있다.



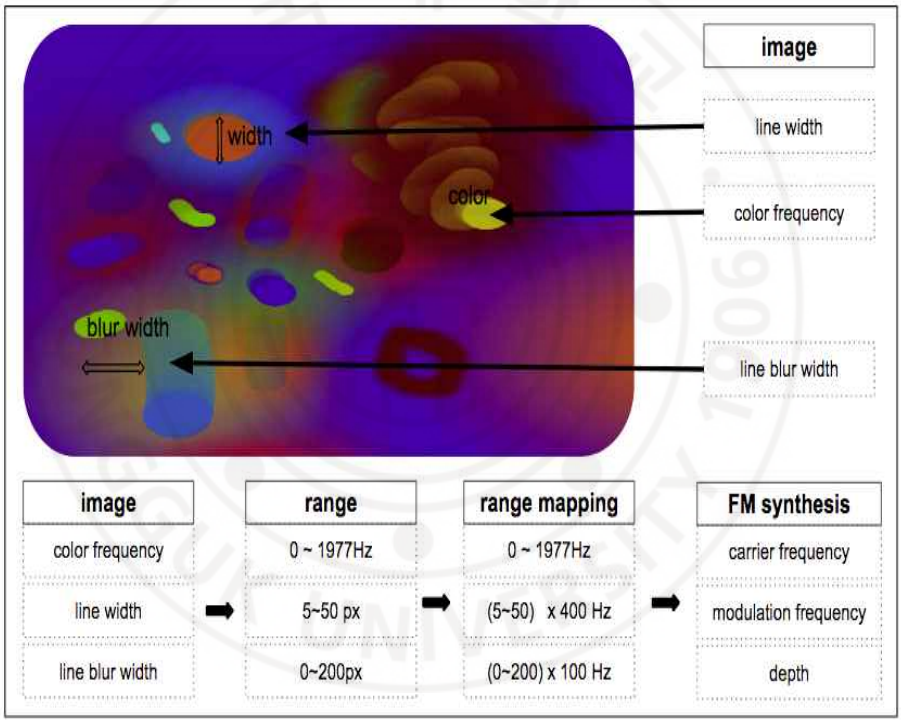
[그림-4.21] 터치 드로잉의 다양한 이미지

메인 캔버스에 그려지는 이미지는 모바일에서 그리는 이미지의 위치 값과 색상 등의 모든 조건 값을 동일하게 사용하여 적용된다. 모바일을 통해 실시간으로 이미지가 그려지면 네트워킹을 통해 메인 화면으로 각각 설정된 데이터가 전달되어 FM 신디사이저를 실행한다. [그림-4.22]는 실시간으로 전달받은 이미지 데이터 값을 통해 FM 신디사이저의 파라미터 값과 속성 값을 실시간으로 제어하는 과정을 보여주고 있다.

```
var fm = new  DSX.FM({
  carrier_type:"sine", carrier:500, mod_type :"sine", modfreq:700
  ,depth:800, gain:0.5 });
var gain = new DSX.Amp({ gain: 0.0});
fm.connect(gain);
gain.connect(DAC);

socket.on('line', function (data) {

  fm.carrier= data.pitch;
  fm.modfreq= data.pitchOne*400;
  fm.depth= data.pitchTwo*100;

  var CarrierDate = DRAW.Framework.FmCarrierType();
  fm.carrier_type= CarrierDate;
  fm.mod_type= CarrierDate;

});
```

[그림-4.22] FM 신디사이저 설정

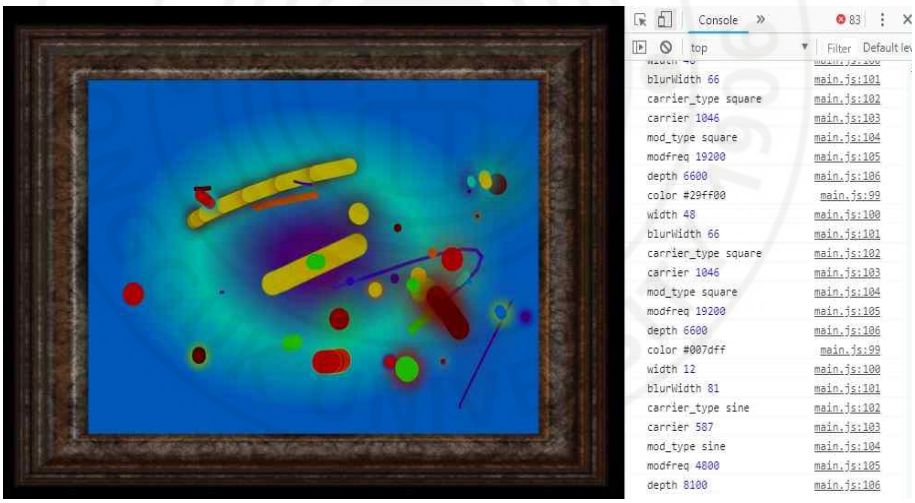
전달받은 이미지 데이터의 색상 주파수 값의 범위는 0~1977Hz로 FM의 carrier frequency 값에 사용되고 있으며 라인의 넓이는 5~50px이고 400값을 곱하여 modulation frequency 값을 제어한다. 블러 넓이는 0~200px이며 100을 곱하여 FM의 depth를 제어한다. [그림-4.23]은 이미지와 사운드 데이터 맵핑의 구성을 보여주고 있다.



[그림-4.23] 이미지와 사운드 데이터 맵핑

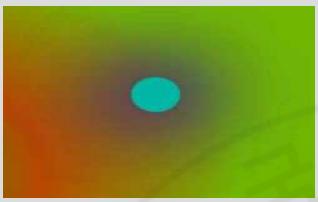
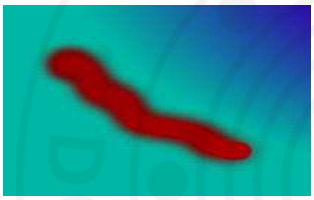
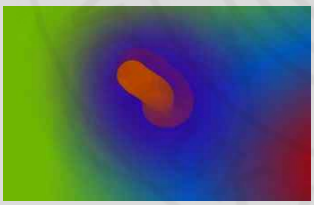
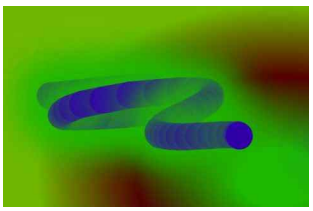
이미지와 사운드에 적용된 맵핑 데이터의 속성 값 확인을 위해 콘솔로그(console.log())를 설정하고 브라우저의 개발자도구를 통하

여 네트워킹 되는 데이터 값을 확인하였다. [그림-4.24]는 콘솔 로그를 통한 이미지와 사운드 데이터 값을 개발자 도구를 통해 보여주고 있다. <표-4.3>은 로그데이터를 통해 이미지 설정 값과 FM 신디사이저의 사운드에 적용된 맵핑 데이터를 나타내고 있다. 첫 번째 이미지의 width 값 43에 맵핑을 통해 설정된 400이 곱해져서 FM 신디사이저의 modfreq 값에 17200이 적용되고 있고 blurWidth 값 164에 10이 곱해져서 depth 값에 16400이 적용되고 있다. FM 신디사이저의 속성 값 carrier_type와 mod_type은 캔버스에 터치 이벤트가 일어날 때 랜덤하게 설정된다.



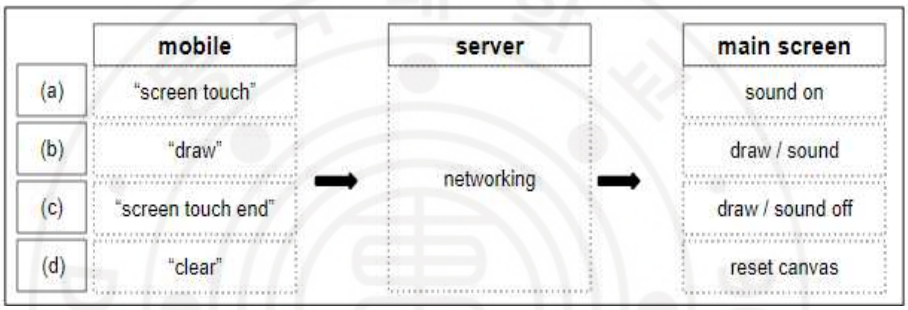
[그림-4.24] 콘솔로그를 통한 이미지와 사운드 데이터 값 모니터

<표-4.3> 이미지와 사운드에 적용된 맵핑 데이터

이미지	데이터 맵핑	
	이미지	FM 신디사이저
	color: #00ffe3 width: 43 blurWidth: 164	carrier_type: sine carrier: 277 mod_type: sine modfreq: 17200 depth: 16400
	color: #ed0000 width: 20 blurWidth 22	carrier_type: sine carrier: 830Hz mod_type: sine modfreq: 8000Hz depth: 2200Hz
	colo:r: #57009f width: 29 blurWidth: 106	carrier_type: square carrier: 698Hz mod_type: square modfreq: 11600Hz depth: 10600Hz
	color: #00ffe3 width: 39 blurWidth: 59	carrier_type: triangle carrier: 277Hz mod_type: triangle modfreq: 15600Hz depth: 5900Hz

(2) 설치작품의 인터랙티브 네트워킹

본 설치작품의 네트워킹 시스템은 Node.js 서버를 통해 모바일 애플리케이션의 이미지 데이터를 메인 화면에 전송하는 흐름으로 구현되었다.



[그림-4.25] 모바일 터치에 따른 네트워킹 순서도

모바일 애플리케이션의 데이터는 서버에 전달되어 콜백(callback) 함수를 통해 실시간으로 메인 화면에 전달되며 모바일 에서 메인 화면에 전송되는 네트워킹 순서는 [그림-4.25]와 같다. [그림-4.25] (a)는 모바일의 캔버스 영역을 터치하면 서버를 통해 메인 화면으로 사운드 음량 데이터를 전송한다. 그 다음 [그림-4.25]의 (b)는 그려지는 이미지 데이터를 서버를 통해 메인 화면에 제공하고 이 데이터를 통해 이미지와 사운드 값을 제어한다. [그림-4.25]의 (c)에서는 터치를 정지하는 순간 메인 화면에 이미지와 사운드 값을 제공하지 않는다. [그림-4.25]의 (d)는 캔버스 이미지를 리셋하는 설정으로 모

든 정보는 초기화되며 메인 화면은 리셋 데이터를 받아 새로운 배경이미지를 생성한다. 위의 네트워킹 순서 흐름에 따라 구현된 각각의 소스코드는 다음과 같다.

```
// mobile control application
  isDown = true;
  oldPoint = new touchPoint(event.targetTouches[0], this);
  socket.emit('start', { start: "start" });

// Node.js server
socket.on('start', function (data) {
  io.sockets.emit('start', data);
});
// main application
socket.on('start', function (data) {
  gain.gain = 0.5;
});
```

[그림-4.26] 모바일 터치 네트워킹 소스코드

모바일 터치에 따른 네트워킹 소스코드는 [그림-4.26]과 같다. 모바일에서 캔버스영역에 터치가 실행되면 Node.js 서버는 정보를 받아 메인 함수로 정보를 전달하게 되고 메인 화면의 콜백 함수는 데이터 정보를 받아 사운드 음량 값을 0.5로 설정한다. 터치가 시작되고 캔버스에 그려지는 이미지의 구성요소 데이터는 객체형태로 전달된다.

```
// mobile control application
socket.emit('draw', {
   width: width, color: color, blurWidth:blurWidth, blurColor:blurColor,
   x1: oldPoint.x, y1: oldPoint.y, x2: newPoint.x, y2: newPoint.y,
   pitch:RANDOMPITCH, pitchOne:RANDOMWIDTH,
   pitchTwo:RANDOMBLUR
});

// Node.js server
socket.on('draw', function (data) {
   io.sockets.emit('line', data);
});

// main application
socket.on('line', function(data) {
   context.lineWidth = data.width;
   context.strokeStyle = data.color;
   context.shadowBlur = data.blurWidth;
   context.shadowColor = data.blurColor;
   context.lineCap = "round";
   context.beginPath();
   context.moveTo(data.x1, data.y1);
   context.lineTo(data.x2, data.y2);
   context.stroke();
});
```

[그림-4.27] 이미지 데이터 객체

[그림-4.27]은 그려지는 이미지 데이터 객체의 네트워킹 프로세스를 나타내고 있다. 모바일을 통해 그려진 이미지 데이터는 객체로 서버에 전달되어 메인화면에 이미지를 생성한다.

```
// mobile control application
  isDown = false;
  oldPoint = new touchPoint(event.targetTouches[0], this);
  socket.emit('end', { end: "end" });

// Node.js server
socket.on('end', function (data) {
  io.sockets.emit('end', data);
});
// main application
socket.on('end', function (data) {
  gain.gain = 0.0;
});
```

[그림-4.28] 모바일 터치 off 네트워킹

[그림-4.28]은 모바일 터치 off의 네트워킹 프로세스를 나타내고 있다. 모바일에서 터치가 off 되면 touchend 이벤트가 발생되면서 off 데이터를 서버에 전달하고 서버는 바로 메인 화면에 off 데이터를 전달한다. off 데이터를 받으면 사운드 음량 값을 0으로 설정하여 사운드를 정지시킨다.

[그림-4.29]는 모바일을 통해 메인 화면을 리셋하는 API 이다. 모바일에서 초기화 정보를 서버로 보내게 되면 메인화면은 데이터를 실시간으로 받아 메인화면의 이미지를 초기화하며 새로운 배경 이미지를 로딩한다.

```
// mobile control application
  context.clearRect(0, 0, canvas.width, canvas.height);
  socket.emit('clear', {
    clear: "clear",
  });

// Node.js server
socket.on('clear', function (data) {
  io.sockets.emit('clear', data);
});
// main application
socket.on('clear', function (data) {
  DRAW.CLEAR = data.clear;
  context.clearRect(0, 0, canvas.width, canvas.height);
  DRAW.Framework.BackGroundSet();
});
```

[그림-4.29] 캔버스 리셋 네트워킹 소스코드

Ⅴ. 성능 고찰 및 비교분석

본 장에서는 본 논문에서 구현한 오디오 라이브러리와 웹 애플리케이션의 성능을 고찰하고 기존 연구와의 기능을 비교분석한다. 이를 위해 다음과 같은 구성으로 연구를 진행하였다.

첫째, 구현된 DrSax.js의 연구 목적에 대한 개념을 정의한다. 그 다음 구현된 기술의 특징과 성능을 분석한 후 Web Audio API, 기존의 라이브러리와 기능을 비교 분석한다.

둘째, DrSax.js를 활용하여 구현된 오디오 공연 시스템 DrWebSax의 성능을 분석하고 실시간 인터랙션 속도를 측정한다.

셋째, DrSax.js를 활용하여 구현된 미디어 설치 작품 Composition 2017의 성능을 분석하고 실시간 인터랙션 속도를 측정한다.

넷째, 앞서 구현된 웹 애플리케이션의 특징을 비교분석하여 본 연구의 타당성을 검증한다.

1. 오디오 라이브러리 성능 고찰 및 비교분석

DrSax.js는 오디오 시스템 구현을 목적으로 연구된 라이브러리이다. 이를 위해 Web Audio API, HTML5, 자바스크립트 등의 웹 기술을 사용하여 인터페이스를 구현하였다. 3장에서 언급된 DrSax.js의 특징을 기준으로 Web Audio API와 비교하여 장단점과 차이점을 고찰한다.

첫째, DrSax.js는 코딩 신택스를 최소화하여 가독성과 프로그래밍의 효율을 증대하고자 하였다. 아래 코드예제는 Web Audio API와 DrSax.js의 오실레이터 설정을 보여주고 있다.

```
// Web Audio API
var DSX = new AudioContext();
var osc = DSX.createOscillator();
var gain = DSX.createGain();
osc.frequency.value = 440;
osc.type = 'square';
gain.gain = 0.5;

// DrSax.js
var DSX = new DSX;
var osc = new DSX.Osc({type:"square", freq: 440});
var gain = new DSX.Amp({gain: 0.5});
```

[그림-5.1] DrSax.js의 코드신택스 비교

DrSax.js의 신택스가 Web Audio API에 비해 간결하여 가독성이 좋다. 오실레이터 인터페이스를 선언하면 웨이브 타입과 주파수 값을 동시에 제어할 수 있다. 간결한 코드 신택스는 유지보수에 이점이 있어 프로그래밍의 효율이 증대된다.

둘째, DrSax.js는 공연에 필요한 오디오 기능 제공을 목적으로 하고 있다. 이에 DrSax.js에서 제공하는 기능이 오디오 공연에 필요한 요소인지 확인하기 위해 먼저 오디오 공연의 개념을 고찰한다.

오디오 공연의 본질은 사운드라는 매개체를 관객에게 전달하는 것이다. 이러한 사운드는 다양한 매체를 통해 공연의 컨셉과 목적을 나타낸다. 오디오 공연에서 사운드는 크게 세 가지 매체로 구분된다. 매체의 유형은 사운드 입력, 전자 사운드, 테잎(tape)음악이며 사운드 매체 요소를 <표-5.1>과 같이 정의하였다.

<표-5.1> 오디오 공연의 사운드 매체

	사운드 매체	사운드 매체 요소
오디오 공연의 사운드 매체	사운드 입력	- 마찰이나 진동을 통한 음향 사운드 - 목소리, 악기
	전자 사운드	- 신디사이저 - 컴퓨터 소리 합성
	테잎 음악	- 노래, 악기, 전자 사운드 등의 사운드 소스를 오디오 파일로 디지털화

DrSax.js는 위에서 정의된 사운드 매체 요소를 인터페이스로 제공하며 그 종류는 <표-5.2>와 같다.

<표-5.2> 사운드 매체와 DrSax.js의 기능 비교

사운드 매체	DrSax.js 인터페이스	기능
사운드 입력	sound input - DSX.Mic()	마이크 값 입력
전자 사운드	synthesis - DSX.Osc() - DSX.FM() - DSX.AM() - DSX.Subtract()	전자 소리 합성
테잎 음악	audio player - DSX.musicPlayer() - DSX.BGsound() - DSX.BGdata()	오디오 파일 출력

또한 사운드 매체 요소 이외에 공연에 필요한 사운드 이펙터, 비주얼라이제이션, 녹음기 등의 인터페이스를 제공한다. 특히 이러한 인터페이스는 Web Audio API에서 제공하지 않아 HTML5와 자바스크립트 기술을 활용하여 구현하였으며, 이 인터페이스를 활용하면 신속한 오디오 공연 시스템 개발이 가능하다. 공연에 필요한 기능을 구현하기 위한 설계,

개발, 테스트 등의 복잡성을 단순화하였기 때문이다. 예를 들어 사운드 매체 요소의 사운드 인풋 기능은 Web Audio API의 getUserMedia()함수를 사용하여 쉽게 개발이 가능하다.

```
// Javascript
var getFile_sound = new XMLHttpRequest();
getFile_sound.open("GET", "music.mp3", true);
getFile_sound.responseType = "arraybuffer";
getFile_sound.send();
getFile_sound.onload = function() {
    var audioData = getFile_sound.response;
    drsax.decodeAudioData(audioData, function(buffer) {
        myBuffer = buffer;
    });
};
// DrSax.js
var file = new DSX.BGdata("music.mp3");
```

[그림-5.2] 오디오 플레이어 코드신택스

테잎음악 재생에 필요한 오디오 플레이어는 HTML5의 <audio>를 사용해 구현이 용이하지만 재생 속도 제어 기능을 제공하지 않는다. 이 기능을 사용하기 위해서는 XMLHttpRequest() 등의 자바스크립트 기술을 활용하여 [그림-5.2]의 코드처럼 오디오 플레이어를 개발해야 한다. 이러한 작업은 난이도 있는 자바스크립트 문법을 필요로 하며 개발 과정이

단순하지 않다. 오디오 플레이어 이외에 녹음, 튜너, 비주얼라이제이션 등도 복잡한 개발 과정을 필요로 한다. DrSax.js는 이러한 기능을 인터페이스로 제공하고 있어 공연 시스템 구현에 효율적인 장점을 가지고 있다. 또한 제공되는 오디오 인터페이스는 공연이외에 오디오 시스템 구축에 활용이 가능하다.

셋째, HTML5의 캔버스 API를 활용한 비주얼라이제이션 인터페이스를 제공한다. 이 기능은 공연에 필요한 요소로 오디오의 음색, 음량, 주파수 등을 시각적으로 모니터링 해준다. 예를 들어 공연 대기 중에는 사운드를 출력할 수 없어 마이크 음량 값을 시각적으로 모니터링 해야 한다. 또한 메인 사운드 음량 값도 실시간으로 확인해야한다. 이러한 이유로 비주얼라이제이션 인터페이스 기능을 DrSax.js에 구현하였으며 이는 Web Audio API와 차별화된 특징이다.

넷째, 데이터 컨트롤 인터페이스는 오디오 데이터 제어와 상태변경을 위해 개발하였다. 특히 인터랙티브 공연은 실시간 데이터 처리를 통해 오디오, 영상 등의 요소와 인터랙션을 연출한다. 이에 데이터 컨트롤 인터페이스를 활용하여 인터랙티브 시스템을 구현한다면 간결한 데이터처리가 가능하다. DrSax.js는 Web Audio API에서 제공하지 않는 데이터 컨트롤 인터페이스를 통해 보다 빠른 인터랙티브 시스템 구현 환경을 제공한다.

지금까지 DrSax.js와 Web Audio API와의 특징에 대해 살펴보았다. DrSax.js는 Web Audio API에서 제공하지 않는 오디오 인터페이스와 비주얼라이제이션, 데이터 컨트롤 기능을 제공해 공연에 필요한 기능을 제

공한다. 하지만 Web Audio API를 활용해 구현된 DrSax.js의 인터페이스는 Web Audio API에서 제공하는 오디오기능의 일부만을 활용하여 구현되었으며 최소한의 기능을 제공해 활용 범위의 제약이 있다. 또한 비주얼라이제이션의 경우 2종류의 제한적인 인터페이스를 제공해 추가 연구를 필요로 한다.

다음은 DrSax.js와 앞서 연구된 라이브러리의 특징을 비교분석하였다.

Web Audio API 기반의 오디오라이브러리는 신디사이저, 사운드 이펙터 등의 유사한 기능을 제공하지만 구현 목적에 따라 앞에서 정의한 사운드 매체를 다르게 제공한다. DrSax.js는 기존의 오디오 라이브러리에 비해 오디오 공연에 필요한 기능을 제공해 오디오 시스템 구현에 효율적이다. 이를 토대로 DrSax.js에서 제공하는 오디오 시스템 기능을 비교분석한다.

Audiolib.js는 신디사이저와 이펙터 위주의 기능을 제공하지만 사운드 입력이나 오디오 파일 재생의 기능을 제공하지 않아 공연 사용에 제약사항들이 있다. Audiolib.js의 영향으로 구현된 Gibber는 사운드 입력, 신디사이저, 이펙터 등의 기능을 제공하지만 오디오 재생 기능을 지원하지 않아 테잎음악을 사용하는 공연에 활용되기 어렵다. Tuna는 이펙터 위주의 기능을 제공하고 사운드 매체 요소를 지원하지 않는다. 오디오 공연에 사용하기 위해서는 사운드 매체를 구현하거나 다른 라이브러리를 활용해야한다. 반면 WAAX는 신디사이저와 이펙터, 비주얼라이제이션 등의 기능을 제공해 오디오 공연에 사용가능하다. 하지만 사운드 입력 기능을 제공하지 않아 라이브 악기를 사용한 오디오 공연에 활용되기 어렵다. Gibberish.js 또한 신디사이저와 오디오 플레이어 기능을 제공해 오디오 공연에 활용이 가능하지

만 악기를 사용한 라이브 공연이 어렵다. Interface.js는 비주얼라이제이션과 데이터 제어를 위해 개발되어 다른 라이브러리들과 함께 오디오 공연에 활용될 수 있다.

지금까지 DrSax.js와 기존의 라이브러리와의 특징에 대해 살펴보았다. Web Audio API 기반의 라이브러리는 기본적으로 이펙터와 신디사이저 등의 유사한 기능을 제공한다. DrSax.js는 공연에 필요한 기능을 제공해 빠른 공연 시스템 구현의 장점이 있지만 특정 목적으로 구현되어 확장성이 다른 라이브러리에 비해 제약적이다. 예를 들어 Tuna는 이펙터만 전문적으로 만들어 다른 기능을 제공하지 않지만 DrSax.js에 비해 많은 이펙터를 제공한다. 이는 사용자의 목적에 따라 장점이 될 수 도 있다. 하지만 DrSax.js의 기능을 확장시킨다면 공연에 특화된 라이브러리로 사용될 것으로 기대된다.

2. 인터랙티브 웹 애플리케이션 비교분석

1) 오디오 시스템 성능 분석

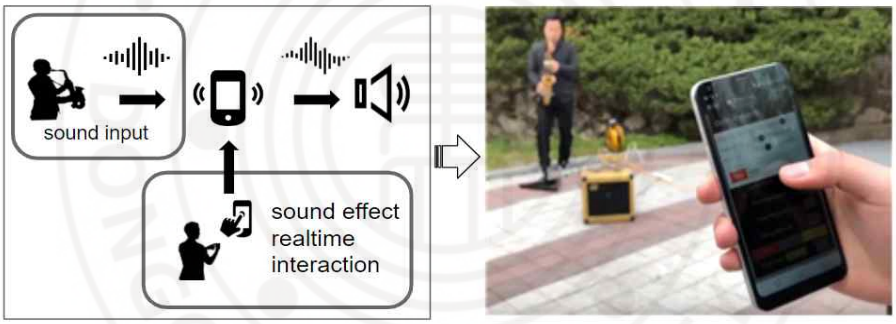
본 항에서는 오디오시스템의 기술 적용을 통해 구현된 DrWebSax의 특징에 대해 살펴보고 DrWebSax와 DrWebSax 컨트롤러의 인터랙션 시간차를 분석한다. 이를 통해 DrWebSax이 실시간 인터랙티브 시스템으로 사용되는데 적합한지 살펴본다.

[그림-5.3] 모바일 DrWebSax의 메인화면

DrWebSax는 2018년, 버스킹(busking)[1] 공연에 사용되었다. 본 공연은 색소폰 연주자와 사운드 디자이너로 구성하였으며 색소폰 연주를 통해

1) 1860년대 영국에서 유래된 것으로 거리에서 자유롭게 공연하는 형태를 의미한다.

사운드 디자이너가 실시간으로 색소폰 사운드를 프로세싱 하는 인터랙티브 멀티미디어 퍼포먼스이다. 본 공연의 시스템 구성은 다음과 같이 간단하게 구성하였다. 마이크입력은 모바일과 PC 모두 내장 마이크를 사용하였고 사운드 아웃풋을 앰프에 연결하여 시스템을 세팅하였다. [그림-5.4]는 모바일 DrWebSax 컨트롤러를 통해 메인 DrWebSax의 사운드를 제어하는 실시간 인터랙션을 보여주고 있다. DrWebSax 컨트롤러의 이펙터 파라미터를 제어하면 실시간으로 메인 DrWebSax의 이펙터 파라미터가 변경되어 사운드의 변화를 준다.



[그림-5.4] 모바일에서의 실시간 데이터 인터랙션

[그림-5.5]의 (a)는 야외 공연모습을 보여주고 있고 (b)는 PC에 로딩된 DrWebSax을 보여주고 있다. [그림-5.5]의 (c)는 공연 전 실시간 인터랙션 테스트 모습이다. 모바일의 이펙터 파라미터를 제어하면 실시간 인터랙션을 통해 DrWebSax의 동일한 이펙터 파라미터가 제어되며 동시에 사운드가 변화된다. [그림-5.5]의 (d)는 공연 중에 실시간 인터랙션을 위해 DrWebSax 컨트롤러의 이펙터 파라미터를 제어하는 모습이다.

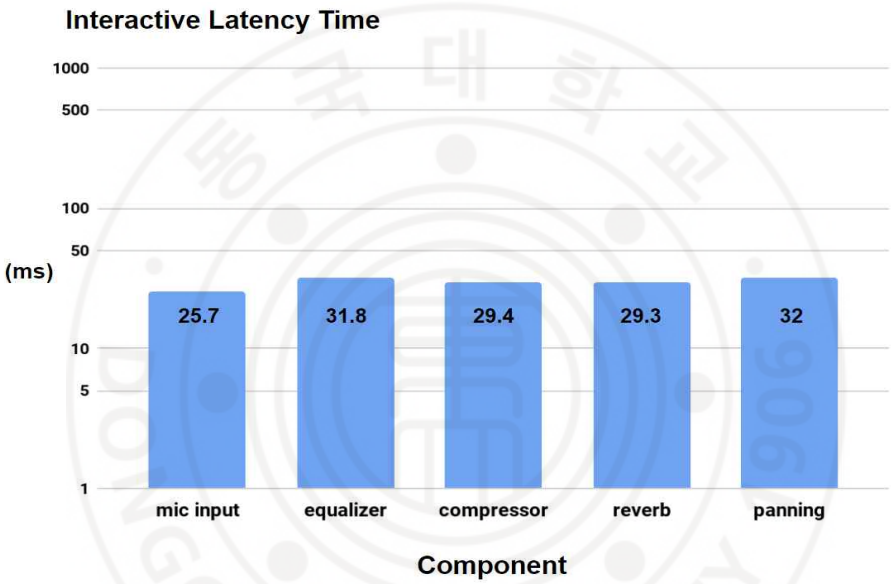
DrWebSax는 PC와 모바일에서 쉽게 실행되어 빠르게 공연 환경을 구축할 수 있다. 특히 모바일의 경우 접근성과 휴대성이 좋아 PC 환경보다 빠른 멀티미디어 공연 시스템 구현이 가능하다.

[그림-5.5] PC DrWebSax와 모바일의 실시간 인터랙션

또한 본 멀티미디어 공연의 실시간 인터랙티브 네트워킹 시스템은 메인 DrWebSax과 DrWebSax 컨트롤러의 이펙터 인터랙션 시간차를 분석하여 효과적으로 구현되었음을 확인하였다. 시간차 분석은 대략 45 Mbps (Mega Bits Per Second) 전송속도의 Wi-Fi 환경에서 다음과 같은 방법으로 진행하였다.

DrWebSax 컨트롤러에서 제어하는 이펙터 파라미터 값과 현재시간을 제공하는 함수 데이터를 메인 DrWebSax에 전달한다. 메인 DrWebSax은 DrWebSax 컨트롤러에서 전송된 시간 데이터와 데이터가 도착한 시간차

를 console.log()를 사용하여 브라우저에 출력한다. 실시간 인터랙션 시간차 분석을 위해 5종류의 이펙터를 테스트하였으며 각 이펙터마다 10번의 파라미터 값을 변경하여 메인 DrWebSax와 DrWebSax 컨트롤러가 인터랙션 되는 시간차 값을 도출하였다.

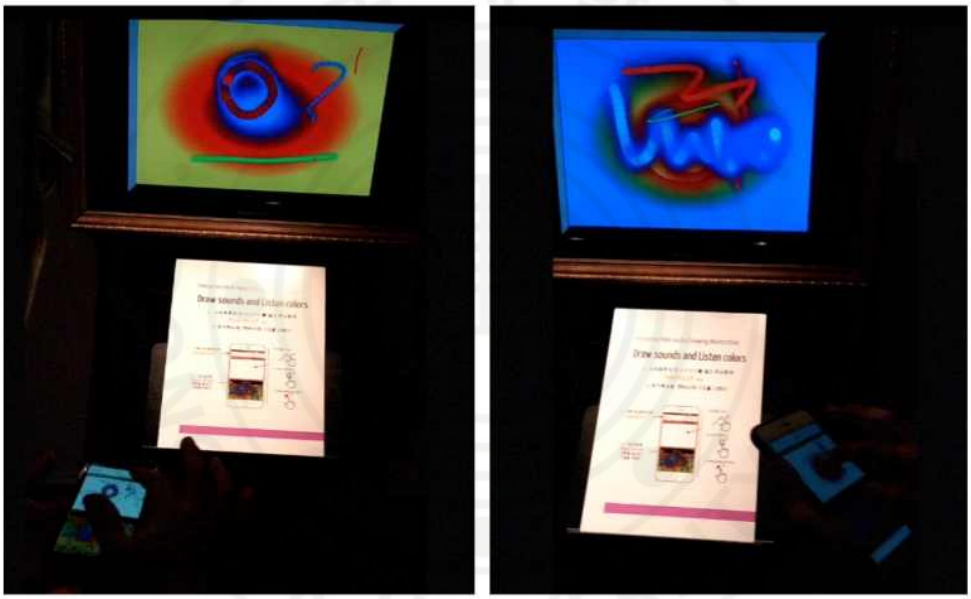


[그림-5.6] DrWebSax의 실시간 인터랙션 시간차 분석

[그림-5.6]은 DrWebSax의 인터랙션에 따른 시간차 평균 결과 값을 그래프로 나타내고 있다. 그래프의 세로 범위 단위는 0~1000ms(millisecond)이며 가로축은 이펙터 컴포넌트를 나타내고 있다. [그림-5.6]의 평균 결과 값을 보면 25.7ms~32ms 사이이며 실시간 인터랙션에 따른 시간차가 거의 없는 것을 알 수 있다. 이를 통해 웹 기반의 실시간 네트워킹 기술이 효과적으로 활용될 수 있음을 검증하였다.

2) 미디어 설치작품의 성능 분석

본 항에서는 DrSax.js와 웹 기술을 적용하여 구현된 미디어 설치 작품 Composition 2017의 특징에 대해 살펴본다. 또한 모바일과 메인 화면의 인터랙션 시간차를 분석하여 실시간 관객 참여가 가능한지 살펴본다.



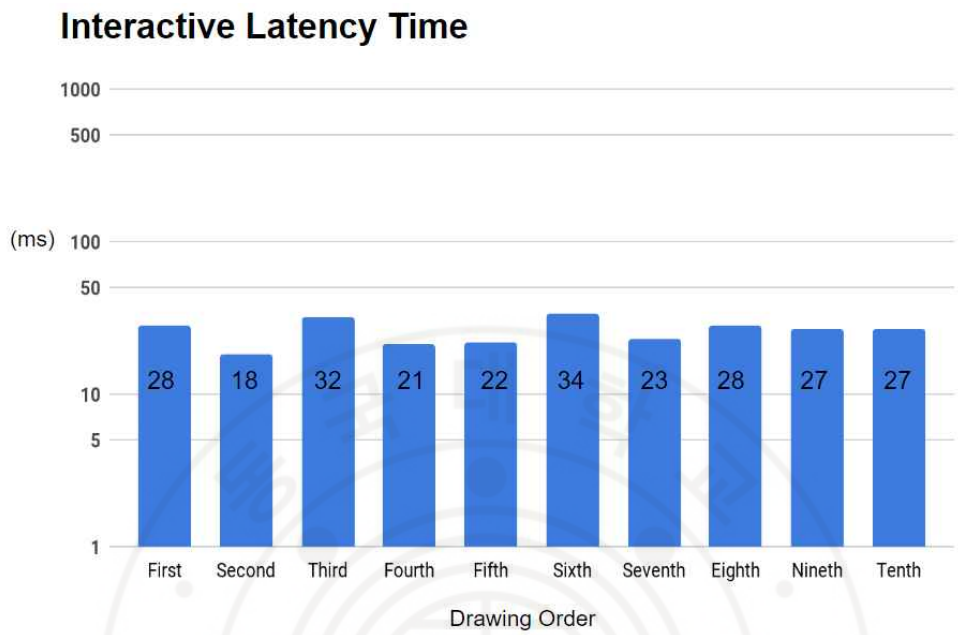
[그림-5.7] Composition 2017의 관객참여 모습

Composition 2017은 DrSax.js의 오디오 시스템과 HTML5의 캔버스 API, Node.js 등의 웹 기술을 활용한 미디어 설치 작품으로 모바일을 통해 관객이 작품에 직접 참여하여 웹 브라우저로 개발된 메인 화면의 이미지와 사운드를 제어한다. [그림-5.7]은 실제 전시에서 관객이 실시간으

로 작품을 구현하는 모습이다. 관객의 모바일에서 그려지는 이미지는 실시간으로 메인 화면에 동기화되며 이미지의 주파수와 넓이 등에 따라 변화하는 FM 신디사이저는 이미지와 인터랙션 되어 공감각을 표현한다. 이를 통해 관객들은 즉흥적인 요소를 실시간으로 공유하고 인터랙션 하여 정형화되지 않은 하나의 작품을 완성할 수 있다. 또한 모바일과 메인 화면의 이미지 인터랙션 시간차를 분석하여 실시간 인터랙티브 네트워킹 시스템이 효과적으로 구현되었음을 확인하였다. 시간차 분석은 대략 45 Mbps 속도의 Wi-Fi 환경에서 다음과 같은 방법으로 진행 결과를 도출하였다.

모바일에서 이미지를 그린 후, 이미지 데이터와 현재시간을 제공하는 함수 데이터를 같이 메인화면에 전달한다. 메인 화면은 모바일에서 전송된 시간 데이터와 데이터 도착 시간차를 console.log()를 사용하여 브라우저에 출력한다. 실시간 인터랙션 시간차 분석을 위해 모바일에서 10번의 터치를 통해 이미지 데이터와 메인 화면 이미지가 인터랙션 되는 시간차 값을 도출하였다.

[그림-5.8]은 Composition 2017의 인터랙션에 따른 시간차 결과 값을 나타내고 있다. 그래프의 세로 범위 단위는 0~1000ms 이며 가로축은 10번의 모바일 테스트 순번을 나타내고 있다. [그림-5.8]에서 10번의 인터랙션 시간 측정 결과를 보면 18ms~32ms 사이 이며 평균값은 26ms 으로 실시간 인터랙션에 따른 시간차가 거의 없는 것을 알 수 있다. 이를 통해 웹 기반의 실시간 인터랙티브 네트워킹 시스템이 효과적으로 활용될 수 있음을 확인하였다.



[그림-5.8] Composition 2017의 실시간 인터랙션 시간차 분석

3) 웹 애플리케이션 비교분석

본 논문에서는 설치형 애플리케이션으로 구현된 인터랙티브 멀티미디어 퍼포먼스의 공연 재연과 시스템 활용에 대한 제한점을 제시하였다. 이의 대안으로 웹 기술을 활용한 인터랙티브 멀티미디어 시스템을 제안하였다. 이에 본 연구를 통해 개발한 웹 애플리케이션의 효율성을 검증하기 위해 본 연구자가 멀티미디어 공연 “혼(魂)”에 사용한 설치형 애플리케이션 Max/MSP와 인터랙티브 웹 애플리케이션 DrWebSax의 특징을 비교분석한다. 비교 범위는 공연 재연과 시스템 활용에 대한 범주로 다음과 같다.

- 운영체제에 따른 애플리케이션 호환성
- 공연 시스템의 휴대성
- 시스템 환경 구축 및 접근성
- 네트워킹 설정
- 애플리케이션 버전의 소스 호환

이를 통해 DrWebSax이 설치형 애플리케이션에 비해 공연 재연과 시스템 활용에 효과적임을 확인한다.

Max/MSP는 Microsoft Windows[2]와 Mac OS[3] 운영체제가 설치된 컴퓨터에서 사용된다. 반면 웹 애플리케이션의 운영체제는 Microsoft Windows,

2) 마이크로소프트에서 개발한 운영체제.
3) 애플에서 개발한 운영체제.

Mac OS, Android[4], iOS이며 컴퓨터와, 모바일, iPod Touch[5], iPad[6] 등의 스마트기기의 웹 브라우저를 통해 실행된다. 사용 가능한 운영체제가 다양하며, 특히 모바일과 스마트 기기는 컴퓨터에 비해 휴대성의 이점이 있다. 또한 Max/MSP는 새로운 컴퓨터나 다른 운영체제에서 사용할 경우 재설치의 번거로움이 있어 시스템 활용에 제약이 있지만 웹 애플리케이션은 컴퓨터나 모바일의 웹 브라우저로 사용이 가능해 접근성이 좋다.

실시간 인터랙션 시스템은 네트워크 통신 기술을 활용하여 구현된다. Max/MSP는 UDP(User Datagram Protocol)[7], OSC(Open Sound Control)[8] 등의 기술을 활용하여 네트워크 통신이 가능하지만 웹은 URL을 통해 네트워킹 통신 기술을 제공하고 있어 별도의 작업이 필요 없고 여러 사용자가 동시 접속이 가능해 인터랙티브 시스템 활용에 용이하다. 또한 웹 애플리케이션 개발에 필요한 서버, 호스팅 등을 무료로 제공하는 서비스가 많아 유료 서비스인 Max/MSP에 비해 비용부담이 적다. Max/MSP는 버전 업데이트에 따른 추가 비용을 부담해야하며 버전에 따른 소스 호환에 오류를 발생하기도 한다. 웹 브라우저는 업데이트 비용이 따르지 않으며 웹 표준이 제정되어 소스 호환에 장점이 있다. 이와 같은 Max/MSP와 웹 애플리케이션의 특징을 <표-5.3>으로 정리하였다.

---

4) 구글에서 개발한 모바일 기반의 운영체제.
5) 애플에서 개발한 미디어 플레이어로 아이폰의 전화 문자 등의 기능을 제거한 기기.
6) 애플에서 개발한 태블릿 컴퓨터.
7) 데이터를 데이터그램단위로 처리하는 단방향 인터넷 프로토콜.
8) 사운드관련 데이터를 전송하는 플랫폼.

<표-5.3> 설치형 애플리케이션과 웹 애플리케이션 비교분석

	Max/MSP	웹 애플리케이션
지원 운영체제	Microsoft Windows, Mac OS	Microsoft Windows, Mac OS, Android, iOS
지원 기기	컴퓨터	컴퓨터, 모바일, iPod, iPad 등의 스마트기기
휴대성	제한적	모바일, 스마트기기 등의 이점
시스템 접근성	컴퓨터, 운영체제 변경 시 재설치	웹 브라우저 접속
네트워크 방법	UDP, OSC	URL
애플리케이션 비용	유료	유료, 무료
업데이트 방식	유료	무료
소스 호환	버전에 따라 오류 발생 가능	웹 표준이 제정되어 좋음

이를 토대로 앞서 본 연구자가 버스킹 공연에 사용한 DrWebSax는 모바일과 노트북 버전으로 시스템을 구축하였으며 휴대성이 좋아 공연위치이동이 용이하였고 장비의 부담이 없었다. 이에 시스템 환경 구축 시간이 짧아 효율적이었다. 또한 모바일과 노트북으로 쉽게 사용이 가능하여 접근성에 이점이 있었다.

실시간 인터랙션 시스템은 URL 도메인을 사용하여 DrWebSax 애플리케이션을 로딩하였고 네트워킹 설정이 필요 없어 빠른 시스템 구축이 가능하였다. 또한 DrWebSax는 무료 호스팅서비스를 이용하여 Max/MSP에 비해 개발 비용이 절감되었고 개발 과정에서 소스 수정 및 호환에 대한 오류나 버그는 없었다.

지금까지 실제 공연에 사용된 DrWebSax에 대해 살펴보았다. 이를 통해 DrWebSax이 설치형 애플리케이션인 Max/MSP에 비해 공연 재연과 시스템 활용에 효과적임을 확인하였다. 하지만 Max/MSP에서 제공하는 오디오, 영상 기술들은 인터랙티브 시스템을 구축하는데 최적화되어 있다. 이와 같은 Max/MSP 기술들을 웹 기반으로 구현하여 인터랙티브 시스템에 활용하기 위해 보다 많은 연구와 개발이 필요하다.

Ⅵ. 결론

HTML5의 표준화와 더불어 끊임없이 발전하고 있는 웹의 다양한 기술은 실시간으로 음악과 영상, 이미지, 텍스트 등의 다양한 멀티미디어 콘텐츠를 제어하는 인터랙티브 웹 애플리케이션 구현을 가능하게 하였다. 이러한 웹 기반으로 구현되는 애플리케이션은 설치형 애플리케이션에 비해 접근성과 호환성 등의 이점을 가지고 있다.

본 논문은 이러한 웹 기술을 활용하여 오디오 라이브러리를 개발하고 이를 통해 공연과 설치작품 등 멀티미디어 시스템에 사용가능한 인터랙티브 웹 애플리케이션 구현을 목적으로 하고 있다. 이를 위해 크게 4가지 주제로 구분하여 연구를 진행하였다.

첫 번째로 개발된 DrSax.js는 자바스크립트 언어를 사용하여 개발된 Web Auido API 기반의 오디오 라이브러리이다. 사운드 프로세싱과 이펙터, 사운드 합성, 오디오 파일 재생, 비주얼라이제이션 등의 인터페이스를 통해 다양한 오디오 애플리케이션 개발이 가능하며 사용자가 보다 빠르고 쉽게 애플리케이션을 제작할 수 있도록 튜토리얼 사이트와 데모 애플리케이션을 제공한다. 데모 애플리케이션은 사용자가 쉽게 라이브러리를 활용할 수 있도록 실제로 적용된 소스코드를 UI에 첨부하였으며 일부 애플리케이션은 프리셋 기능을 포함하고 있다.

DrSax.js의 인터페이스는 간결한 코드 신택스로 구성되어 있다. 이러한 특징은 가독성이 좋아 빠르고 간결하게 오디오 시스템을 구현할 수

있는 장점을 가지고 있으며 자바스크립트 문법과 오디오 프로세싱에 익숙하지 않는 사용자에게 어렵지 않은 사용 환경을 제공한다.

두 번째로 개발한 DrEditor는 웹 오디오 애플리케이션 개발을 위해 사용자가 쉽게 웹 오디오 코드 테스트와 디버깅을 할 수 있는 에디터이다. DrSax.js의 구성요소로 구현된 프리셋 소스코드를 제공한다.

세 번째로 개발된 DrWebSax는 공연에 사용할 목적으로 설계된 오디오 시스템 애플리케이션이다. DrWebSax는 사운드 프로세싱과 이펙터 제어 등의 연주에 필요한 여러 가지 기능을 탑재하고 있으며 DrWebSax 컨트롤러를 통해 실시간으로 사운드 제어가 가능하다. DrWebSax는 DrSax.js의 라이브러리 기술을 최대한 활용하여 개발하였으며 오디오 컴포넌트에 따라 프리셋을 제공한다. Node.js의 Socket.io 모듈을 통해 실시간 데이터 인터랙션을 구현하였다. 이러한 DrWebSax는 실제 인터랙티브 멀티미디어 공연에 사용되어 인터랙티브 웹 애플리케이션의 가능성을 확인하였고 실시간 인터랙션 동기화 시간차를 측정하여 인터랙션 기술이 효과적으로 적용되었음을 검증하였다.

마지막으로 Composition 2017은 관객이 웹과 모바일을 통해 실시간으로 작품에 참여하고 또 다른 관객과의 인터랙션을 통해 작품을 공유하는 즉, 협업을 통해 작품을 만들어가는 인터랙티브 미디어 설치 작품이다. 관객들은 모바일을 통해 이미지를 그리고 그 이미지는 실시간 인터랙션을 통해 메인 화면에 동기화되고 사운드를 제어한다. 본 설치작품 역시 실시간 인터랙션 동기화 시간차를 측정하여 인터랙션 기술이 효과적으로 적용되었음을 확인하였다.

이와 같이 개발된 인터랙티브 웹 애플리케이션은 오디오 환경 구축을 위해 DrSax.js의 기술을 적용하였다. 간결한 코딩 신택스를 통해 명료하고 가독성 좋은 코드로 설계가 가능하였고 오디오 이펙터와 사운드 프로세싱 시스템을 손쉽게 구현하였다. 또한 오디오 시스템 제어를 위해 개발된 비주얼라이제이션과 데이터 제어 기능이 효과적으로 사용되어 DrSax.js의 기술 활용의 효율성을 확인하였다. 이러한 DrSax.js는 일반 오픈소스로 공개되어 프로그래밍에 익숙하지 않은 사용자부터 고급 사용자까지 다양한 애플리케이션을 구현할 수 있는 오디오 라이브러리로 사용될 수 있을 것이다.

본 논문에서는 Web Audio API를 활용하여 개발된 오디오 라이브러리 DrSax.js를 통해 보다 빠르고 효과적인 인터랙티브 웹 애플리케이션 개발을 구현하였고 성능을 분석하여 웹 기반의 인터랙티브 멀티미디어 콘텐츠 구현의 가능성을 제시하였다.

하지만 앞서 제안된 연구가 인터랙티브 멀티미디어 시스템뿐만 아니라 다양한 웹 오디오 애플리케이션 개발에 활용되기 위해서는 여러 측면에서 향후 연구가 이루어져야한다.

첫째, 라이브러리의 활용도와 확장성에 관련된 것이다. DrSax.js는 공연이나 설치작품 등에 사용가능한 웹 애플리케이션을 빠르게 구현하기 위해 간결한 코드 신택스로 개발되었다. 최소한의 파라미터와 사운드 프로세싱 위주의 인터페이스는 복잡한 DAW나 오디오 편집 애플리케이션에 사용되기에는 제한적이다. 이러한 애플리케이션을 구현하기 위해서는 멀티트랙과 사운드 인풋/아웃풋 채널, MIDI, 오디오 파일 에디팅 등의

관련연구가 선행되어야 한다. 이 연구를 통해 라이브러리의 기능이 확장되면 웹 기반으로 DAW 구현이 가능해져 실시간 협업이 가능해지고 접근성의 이점으로 신속한 오디오 작업이 기대된다.

둘째, 웹 기반의 인터랙티브 오디오 시스템은 복잡한 프로그래밍 코드와 오디오데이터, 네트워킹작업을 실시간으로 처리하기 때문에 최적화된 코드 설계를 필요로 한다. 효율이 좋지 않은 코드로 구현될 경우 메모리 누수로 애플리케이션의 처리 속도와 성능이 저하되며 가독성이 좋지 않아 유지보수에 큰 어려움이 발생한다. 이러한 대안으로 많은 연구와 테스트를 통해 최적화된 코드 기법과 디자인 패턴을 프레임워크로 제공한다면 보다 안정적이고 좋은 성능의 인터랙티브 웹 애플리케이션 개발이 가능할 것이다. 이를 통해 향후 연구에서는 오디오 인터페이스 기능이 확장된 라이브러리 구성과 코드 최적화를 위한 분석 및 테스트가 필요하다. 이러한 인터랙티브 멀티미디어 시스템의 지속적인 연구는 여러 분야의 기술과 융합되어 새로운 웹 기반의 멀티미디어 플랫폼으로 발전할 것으로 기대된다.

Keyword (검색어):

웹 오디오 라이브러리(Web Audio Library), 인터랙티브 웹 애플리케이션(Interactive Web Application), 웹 에디터(Web Editor), 인터랙티브 웹 오디오 시스템(Interactive Web Audio System), 웹 미디어 설치 시스템(Web Media Installation), DrSax.js, DrEditor, Composition 2017, DrWebSax, Web Audio API, 자바스크립트(Javascript), Node.js

E-mail: antaresax@gmail.com

참 고 문 헌

[1] Time. http://time.com/4835243/facebook-2-billion-monthly-active-users/ (visited on 07/12/2018).

[2] Boyd, D. M., &Ellison, N. B. (2007). Social network sites: Definition, history, and scholarship. Journal of computer mediated Communication, 13(1), 210-230.

[3] Berners-Lee, T., Dimitroyannis, D., Mallinckrodt, A. J.,&McKay, S. (1994). World Wide Web. Computers in Physics.

[4] Zikopoulos, P., &Eaton, C. (2011). Understanding big data: Analytics for enterprise class hadoop and streaming data. McGraw-Hill Osborne Media.

[5] Gubbi, J., Buyya, R., Marusic, S., &Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, 29(7), 1645-1660.

[6] Nilsson, N. J. (2014). Principles of artificial intelligence. Morgan Kaufmann.

[7] Li, G., Hou, Y., &Wu, A. (2017). Fourth Industrial Revolution: technological drivers, impacts and coping methods. Chinese Geographical Science, 27(4), 626-637.

[8] W3C. https://www.w3.org/
(visited on 07/12/2018).

[9] Hickson, I., &Hyatt, D. (2011). Html5. W3C Working Draft WD-html5-20110525, May.

[10] Chrome. https://www.google.com/intl/en_ca/chrome/ (visited on 07/12/2018).

[11] Frain, B. (2012). Responsive web design with HTML5 and CSS3. Packt Publishing Ltd.

[12] JavaScript. https://developer.mozilla.org/en-US/docs/Learn/JavaScript (visited on 07/12/2018).

[13] Node.js. https://nodejs.org (visited on 07/12/2018).

[14] Node.js. https://developer.mozilla.org/en-US/docs/Glossary/Node.js (visited on 07/12/2018).

[15] Andreessen, M., &Bina, E. (2010). NCSA Mosaic: a global hypermedia system. Internet Research, 20(4), 472-487.

[16] Schatz, B. R., &Hardin, J. B. (1994). NCSA Mosaic and the World Wide Web: global hypermedia protocols for the Internet. Science, 265(5174), 895-901.

[17] Lamm, S. E., Reed, D. A., &Scullin, W. H. (1996). Real-time geographic visualization of World Wide Web traffic. Computer Networks and ISDN Systems, 28(7-11), 1457-1468.

[18] https://html.com/attributes/audio-controls/ (visited on 11/06/2019).

[19] https://developer.mozilla.org/ko/docs/Web/HTML/Element/audio (visited on 11/06/2019).

[20] Jot, J. M. (1999). Real-time spatial processing of sounds for music, multimedia and interactive human-computer interfaces. Multimedia systems, 7(1), 55-69.

[21] Camurri, A., &Ferrentino, P. (1999). Interactive environments for music and multimedia. Multimedia systems, 7(1), 32-47.

[22] Microsoft. https://www.microsoft.com/en-us/. (visited on 11/06/2019).

[23] Kimber, D., &Wilcox, L. (1997). Acoustic segmentation for audio browsers. Computing Science and Statistics, 295-304.

[24] Didkovsky, N., &Hajdu, G. (2008). Maxscore: Music Notation in Max/MSP. In ICMC.

[25] Eigenfeldt, A. (2007). Drum Circle: Intelligent Agents in Max/MSP. In ICMC.

[26] Kraft, S., &Zölzer, U. (2014, May). BeaqleJS: HTML5 and JavaScript based framework for the subjective evaluation of audio quality. In Linux Audio Conference, Karlsruhe, DE.

[27] Web Audio API. https://developer.mozilla.org/ko/docs/Web/API/Web_Audio_API (visited on 07/12/2018).

[28] Web Audio API. https://www.w3.org/TR/webaudio/ (visited on 07/12/2018).

[29] HTML5 browser. http://html5test.com/results/desktop.html (visited on 07/12/2018).

[30] Yan, X., Yang, L., Lan, S., &Tong, X. (2012, August). Application of HTML5 multimedia. In 2012 International Conference on Computer Science and Information Processing.

[31] WebGL. for the web. https://www.khronos.org/webgl/
(visited on 07/12/2018).

[32] Audiolib.js. https://github.com/jussi-kalliokoski/audiolib.js/
(visited on 07/12/2018).

[33] C. Roberts and J. Kuchera-Morin. Gibber : Live coding audio in the browser. In Proceedings of the International Computer Music Conference (ICMC 2012), 2012.

[34] Tuna.js. https://github.com/Theodeus/tuna
(visited on 07/12/2018).

[35] H. Choi and J. Berger. WAAX : Web audio api extension. in Proceedings of the 13th Conference on New Interfaces for Musical Expression(NIME-13), Daejeon, Korea, 2013.

[36] Hongchan Choi. “Collaborative musicking on the web”, 2016
https://searchworks.stanford.edu/view/11685327
(visited on 07/12/2018).

[37] C. Roberts, G. Wakefield, and M. Wright. The web browser as synthesizer and interface. In Proceedings of the New Interfaces for Musical Expression conference, 2013.

[38] Crockford, Douglas(208). JavaScript : TheGodParts, O'Reily.

[39] Crockford, Douglas(2010). JavaScript Patterns : Build Better Applications with Coding and Design Patterns, O'Reily.

[40] Nicholas C. Zakas(2014).The Principles of Object-Oriented JavaScriptO'Reily.

[41] Tutorialpoints. https://www.tutorialspoint.com/
(visited on 07/12/2018).

[42] Play the light of monet. https://youtu.be/uZLOY8onwz4
(visited on 07/12/2018).

[43] JSFiddle. https://jsfiddle.net/
(visited on 07/12/2018).

[44] CodePen. https://codepen.io/
(visited on 07/12/2018).

[45] W3Schools. https://www.w3schools.com/
(visited on 07/12/2018).

[46] JavaScript. Weekly, https://javascriptweekly.com/
(visited on 07/12/2018).

[47] Bootstrap. https://getbootstrap.com/docs/4.0/getting-started/javascript/
(visited on 07/12/2018).

[48] JavaScript. https://developer.mozilla.org/ko/docs/Web/JavaScript
(visited on 07/12/2018).

[49] JavaScript API. https://developers.google.com/api-client-library/
javascript/start/start-js (visited on 07/12/2018).

[50] 2009 Resonation. https://youtu.be/b3RpjOUMdwg
(visited on 07/12/2018).

[51] McCahill, M. (1994). Uniform Resource Locators (URL).

[52] Lawson, B., &Sharp, R. (2011). Introducing html5. New Riders.

[53] Mogul, J. C., Gettys, J., Berners-Lee, T., &Frystyk, H. (1997). Hypertext Transfer Protocol--HTTP/1.1.

[54] Van Kesteren, A., &Jackson, D. (2007). The xmlhttprequest object. World Wide Web Consortium, Working Draft WD-XMLHttpRequest-20070618, 72.

[55] Rowell, E. (2011). HTML5 Canvas Cookbook. Packt Publishing Ltd.

[56] Oliphant, T. E. (2007). Python for scientific computing. Computing in Science &Engineering, 9(3), 10-20.

[57] Wang, G., Cook, P. R., &Salazar, S. (2015). Chuck: A strongly timed computer music language. Computer Music Journal, 39(4), 10-29.

[58] https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats (visited on 06/03/2018).

[59] Hindle, A. (2013). SWARMED: Captive Portals, Mobile Devices, and Audience Participation in Multi-User Music Performance. In NIME (pp. 174-179).

[60] Essl, G., &Muller, A. (2010, June). Designing Mobile Musical Instruments and Environments with urMus. In NIME (pp.76-81).

[61] Roberts, C., Wakefield, G., Wright, M., & Kuchera-Morin, J. (2015). Designing musical instruments for the browser. Computer Music Journal, 39(1), 27-40.

[62] N. Weitzner, J. Freeman, S. Garrett, and Y. Chen. massMobile - an audience participation framework. Proceedings of the New Interfaces For Musical Expression Conference, 2012.

[63] Lee, S.W. and Freeman, J. Echobo : A Mobile Music Instrument Designed for Audience To Play. In Proceedings of the International Conference on New Interfaces for Musical Expression (NIME). 2013. Daejeon, Korea.

[64] Freeman, J. "Large Audience Participation, Technology, and Orchestral Performance." In Proceedings of the International Computer Music Conference (ICMC). Barcelona, 2005.

[65] Dahl, L., Herrera, J., and Wilkerson, C. "TweetDreams : Making Music with the Audience and the World Using Real-Time Twitter Data." In Proceedings of the International Conference on New Interfaces for Musical Expression (NIME). Oslo, 2011.

[66] Bianciardi, D., Igoe, T., and Singer, E. "Eos Pods : Wireless Devices for Interactive Musical Performance." In Proceedings of the Fifth Annual Conference on Ubiquitous Computing. Seattle, 2003.

[67] Essl, G. and Rohs, M. “Interactivity for Mobile Music-Making.” Organised Sound. 14, no. 2 (2009): 197-207.

[68] N. J. Bryan, J. Herrera, J. Oh, and G. Wang. Momu: A mobile music toolkit. In Proceedings of the International Conference on New Interfaces for Musical Expression (NIME), Sydney, Australia, 2010.

[69] G. Wang. Designing smule’s iphone ocarina. In In Proceedings of the International Conference on New Interfaces for Musical Expression, Pittsburgh, 2009.

[70] Rohs, M. and Essl, G. “Camus2 - Collaborative Music Perfor mance with Mobile Camera Phones.” In Proceedings of the International Conference on Advances in Computer Entertainment Technology (ACE). Salzburg, 2007.

[71] J. Oh, J. Herrera, N. J. Bryan, L. Dahl, and G. Wang. Evolving mobile phone orchestra. In Proceedings of the International Conference on New Instruments for Musical Expression, June 2010.

[72] Hugh Rawlinson, Nevo Segal, and Jakub Fiala. Meyda: an audio feature extraction library for the web audio api. In The 1st Web Audio Conference (WAC). Paris, Fr, 2015.

[73] E. Hong and J. Kim. DrSax.js : a JavaScript based Unified Web Audio Library and Framework. ICACS 2017, Jeju, Korea.

[74] E. Hong and J. Kim. Interactive Web Audio Networking System and Method with DrSax.js. ICCIP 2017, Nov, Tokyo, Japan.

[75] Glimmer. http://leoalmanac.org/journal/Vol_15/lea_v15_n11_12/JFreeman.asp (visited on 14/04/2019).

ABSTRACT

A study of Interactive Web Application through Web-based Audio Library Development

EUY SHICK, HONG
Department of Multimedia
Graduate School of Digital Image and Contents
Dongguk University

Style of music has constantly been developed by musicians and information technologies. Especially, computer music has studied for synthesizing and audio processing on the web technologies with javascript that is a flexible language with a front and server-side capabilities, commonly, working dynamic interactions and networking.

This paper aims an interactive web application development with a web audio library DrSax.js through the Web Audio API and can be

separated into four parts.

First, DrSax.js has developed as a web audio library, making it possible to speedy and easily make web applications to control interactive audio processing, sound effects, visualization, and media art. Normally, web audio applications need data to control UI and visualization, such as on/off buttons, dials, slides. So DrSax.js offers a number of functionalities that visualization and data control function.

Second, this paper supports web code editor and debugging an application that is DrEditor supports variously preset for test and development.

Third, DrWebSax is a multi-sound workstation for saxophone. It works that sound effects, processing, recorder, tuner for live interactive performance in real time.

Last, Composition 2017 is a media art installation that is interactive web audio system using DrSax.js and Node.js with PC and mobile. A user can communicate and control images and sounds processing through other user's application in real-time at various place. Also, users can access the web application with Wi-Fi or 3G, 4G data.

This paper describes a web audio library and interactive web-audio system to control that audio processing, sound effects, visualization.

This paper aims a faster and more effective interactive web application development through the audio library DrSax.js, which was developed using web technology for interactive multimedia contents.

부록 : 본 연구 결과의 소스 코드 및 애플리케이션 URL

1. DrSax.js

   Tutorial site : https://drsax.github.io/DrSAX/lib.1.8.html

   Source Code : https://github.com/drwebsax/DrSax.js

2. DrEditor

   App : http://antaresax.cafe24.com/editor/boots.html

   Source Code : https://github.com/drwebsax/drEditor

3. DrWebSax

   App : 1) main application

   https://webeffect.herokuapp.com/

   2) controller

   https://webeffect.herokuapp.com/cont.html

   Source Code : https://github.com/drwebsax/DrWebSax

4. Composition 2017

   App : 1) main screen

   https://drpaint.herokuapp.com/main.html

   2) controller

   https://drpaint.herokuapp.com/canvas.html

   Source Code : https://github.com/drwebsax/composition2017/