creative commons
COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

석 사 학 위 논 문

노이즈 알고리즘에 기반한 오디오비주얼 인터랙션 시스템 디자인 연구 (멀티미디어 작품 <Event Horizon II>를 중심으로)

지도교수 김 준

동국대학교 영상대학원 멀티미디어학과

오 희 원

2022

석 사 학 위 논 문

노이즈 알고리즘에 기반한 오디오비주얼 인터랙션 시스템 디자인 연구 (멀티미디어 작품 <Event Horizon II>를 중심으로)

오 희 원

지도교수 김 준

이 논문을 석사학위논문으로 제출함

2021 년 12 월

오희원의 음악석사(컴퓨터음악) 학위 논문을 인준함

2022 년 1 월

위원장 정 진 헌

위 원 김 정 호

위 원 김 준

동국대학교 영상대학원

목 차

표 목 차

그 림 목 차

I. 서 론

1. 연구의 배경

오디오비주얼(audio-visual)아트 또는 비주얼 뮤직(visual music)은 음악을 시각적인 형태로 표현하는 작품을 의미한다. 20세기 후반부터 기술의 발전으로 인해 실시간으로 시각요소와 음악과의 상호작용을 표현할 수 있게 되었다. 컴퓨터 그래픽스를 활용한 시각화 기법들이 현대 미술과 전자음악 등에서 청각적 경험 또는 시각적 경험만을 전달하는 것이 아니라 다감각적(multisensory)인 경험을 작품에 담기 위해 사용되기 시작했다.

이러한 오디오비주얼 작품들은 근본적으로 데이터 시각화(data visualization) 기법과 음향화(visual music) 기법에 토대를 두고 만들어진다고 볼 수 있다. 데이터 시각화 기법에 가까운 형태의 작품으로는 백남준(b.1932~2006)의 <참여 TV> [그림-1] 와 같은 작품처럼 마이크를 통해 입력되는 실제 오디오 신호의 변화를 추상적인 도형의 움직임 등으로 데이터의 경향을 시각화해서 보여주는 형식이 있다. 음향화에 가까운 형태의 오디오비주얼 작품으로는 디즈니의 <환타지아>(Fantasia)[1]가 대표적으로

---

[1] 《환타지아》(Fantasia)는 1940년에 제작된 월트 디즈니 프로덕션에서 제작하고 RKO 라디오 픽처스에서 배급한 월트 디즈니의 장편 애니메이션 영화이다. 이 영화는 필라델피아 오케스트라와 지휘자 레오폴드 스토코프스키가 함께 제작된 클래식 음악에 맞춰 움직이는 애니메이션을 담았다.

직접적인 데이터 매핑이 아닌 각 악기와 음색에 어울리는 시각적 효과 또는 움직임을 매핑하는 형태의 시각화 기법을 사용하는 작품들을 의미한다.

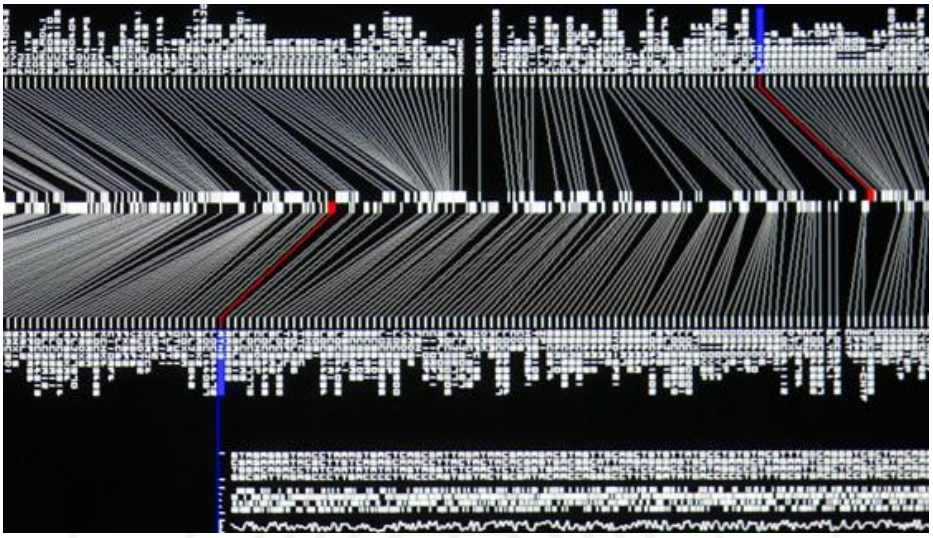
[그림 1] 백남준의 멀티미디어 작품, <참여 TV>[2)]

2000년대 후반 복잡한 음(complex tone)을 구성하는 최소 단위로서의 사인파(sine wave)를 중심으로 디자인된 사운드와, 데이터의 가장 작은 단위인 이진 데이터 (binary data)를 흑(black)과 백(white)의 색상을 가지는 시각 요소로써 표현한 미니멀리즘(minimalism)적 오디오비주얼 작품들이

---

2) 1963년 독일의 부퍼탈에서 열린 백남준의 첫 개인전 《음악의 전시 — 전자 텔레비전》에 등장한 13대의 실험 텔레비전 중 하나이다. 초기 형태의 〈참여 TV〉는 순수하게 청각적인 것에 초점이 맞춰져 있었으나, 이후에 제작된 〈참여 TV〉는 음향의 증폭을 시각화시켜 누군가가 마이크에 대고 소리를 낼 경우 모니터에 여러 이미지가 나타나도록 되어 있다. (백남준 아트센터, https://njp.ggcf.kr/참여-tv/)

료지 이케다(Ryoji Ikeda)[3], 알바 노토(Alva Noto)[4] 등의 음악가들에 의해 발표되고 시도되어왔다.



[그림-2] 료지 이케다의 작품 <Datamatics>의 한 장면

미니멀리즘적 시도를 대표하는 아티스트인 료지 이케다의 'data' 연작 <datamatics>의 설치 작품 [그림-2]를 살펴보면 데이터의 최소 단위인 이진 데이터를 시각화하기 위한 주 재료로 흑색 배경의 백색 타이포그래피 및 바코드, 픽셀레이션 기법 등을 사용하였다. 흑-백과 같은 상반된 시각 요소를 음향화하기 위해 사인파와 노이즈(noise)를 사용한 음향적 대비를 이용한 것을 살펴볼 수 있었으며 이러한 도식화 경향은 료지 이케다의 작업

[3] Ryoji Ikeda, (b. 1966) 는 일본 태생의 비주얼 / 사운드 아티스트이다.

[4] Alva Noto, (b. 1965) 는 Carsten Nicolai의 아티스트명으로, 독일 태생의 전자음악가이다.

뿐만 아니라 알바 노토 & 류이치 사카모토의 연작인 ‘uni‘ 시리즈에서도 찾아볼 수 있었다. 본 연구에서는 이들의 작품들을 미니멀리즘이라는 하나의 흐름으로 보고 이러한 작품에서 반복적으로 사용되는 개념적 요소인 노이즈, 모노크롬(monochrome) 기반의 시각화와 같은 최소한의 구성 요소 및 개념들을 전유하되 미니멀리즘 작품에서 나타나는 사인파-노이즈 도식에 기반한 시각화의 패턴에서 벗어난 새로운 시각화 기법을 찾아보고자 한다.

2. 연구의 목적

본 연구는 두 가지 키워드, 생성 예술(generative art) 과 노이즈(noise)를 중심으로 진행되었다. 생성 예술은 인간이 아닌 자율적인 시스템이 직접 작품의 특징과 형상을 결정짓는 형태의 예술 작품이다. 작업자가 특정한 알고리즘 또는 시스템을 통해 산출해내는 결과물 또는 그 과정. 계획되거나 프로그래밍 된 조정을 통해 어떤 시스템에 의해 예술작품이 만들어지는 작품을 생성 예술이라 부른다.[5)]

노이즈는 잡음이나 임의적 시각적인 자극을 뜻하는 언어로써, 영상에서는 일관된 색채나 형태를 보이지 않는 형태, 음향에서는 비주기성을 띠는 음색을 뜻하는 언어이다. 한편 기술적 요소로써 노이즈는 매체에 따라 서로 다른 무작위적 패턴을 임의로 구현하는 데 사용되어왔다.

컴퓨터 그래픽스에서 노이즈란 자연계에서 보이는 무작위성, 즉 프랙탈 적 형태를 구현하기 위해 이용되는 알고리즘을 의미한다. Perlin noise[6)], simplex noise[7)]와 같은 노이즈 알고리즘은 컴퓨터 그래픽스에서 실시간으로 지형 텍스쳐를 생성하는 기법 (generative terrain) 또는 가상에

---

5) 박상현 (2012). 제너레티브 아트의 미학적 맥락과 기술적 유사 영역에 대한 연구. 디지털디자인학연구, 12(2), 179-189

6) Perlin 노이즈는 1983년 Ken Perlin에 의해 만들어진 난수 발생 알고리즘으로 컴퓨터 그래픽에 절차적(procedural) 텍스처를 주어 자연에서 볼 수 있는 구름이나 풍경, 대리석 같은 패턴이 있는 텍스처를 표현하는데 주로 사용된다.

7) Perlin 노이즈 알고리즘에서 컴퓨팅 연산의 부하를 해소시킨 형태의 알고리즘

존재하는 물체의 질감을 좀 더 현실적인 형태로 만드는 데 사용되어 왔으며 (normal map) 제네러티브 아트로 대표되는 추상적인 예술 작품의 생성 기법으로도 사용되었다.

화이트 노이즈(white noise)[8]로 대표되는 청각적 노이즈는 오디오 필터 및 딜레이를 접목한 물리 모델 기반의 음원 합성에 쓰이거나 디지털 오디오의 해상도 변환 시 생기는 왜곡의 감쇄(dithering)[9]처럼 인위적인 데이터에 추가되어 자연스러운 출력물을 생성하기 위한 재료로 쓰여왔다. 이러한 노이즈의 다양한 적용 사례 중 생성 예술의 주 매체로서의 시각 노이즈와 음원 합성 방식의 주 매체로 사용되는 청각적 노이즈 두가지 재료를 토대로 오디오 스펙트럼 데이터나 트랜지언트(transient)[10] 검출 기술을 접목하여 3 차원 공간상에 음악을 시각화하는 비주얼라이징 시스템을 개발하고자 하는 것이 본 연구의 목적이다.

---

8) 백색 소음, 일정한 청각 패턴 및 주기가 없이 전체적이고 일정한 주파수 스펙트럼을 가진 노이즈를 뜻한다.

9) 디더링은 디지털 오디오의 비트 뎁스를 감쇄하면서 생기는 왜곡을 감쇄시키기 위해 노이즈를 삽입하는 기술이다.

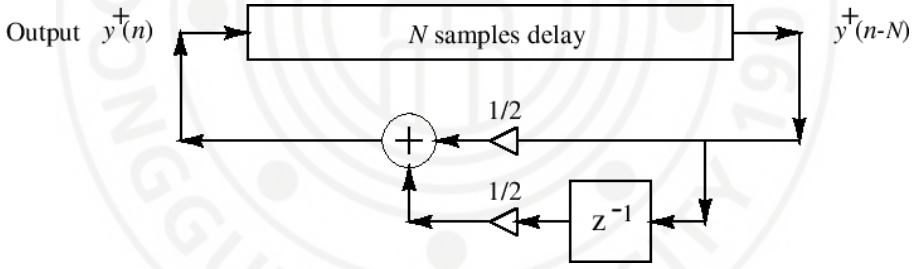
10) 음향에서 트랜지언트(Transient)란 강한 진폭을 가지는 짧은 신호를 뜻한다.

II. 기술 연구

1. 사운드 디자인

1) Karplus-Strong 알고리즘을 사용한 퍼커션 디자인

FeedbackNoise는 본 작품을 위하여 만든 Max for Live[11] 악기로써 Karplus-Strong 알고리즘을 사용한 신디사이저이다. 해당 악기를 작품의 중심 요소인 드럼 악기로 사용하여 IDM[12] 이라는 장르에서 주로 사용되는 리듬 시퀀스를 연출하였다.



[그림-3] Karplus-Strong 알고리즘의 Signal Diagram[13]

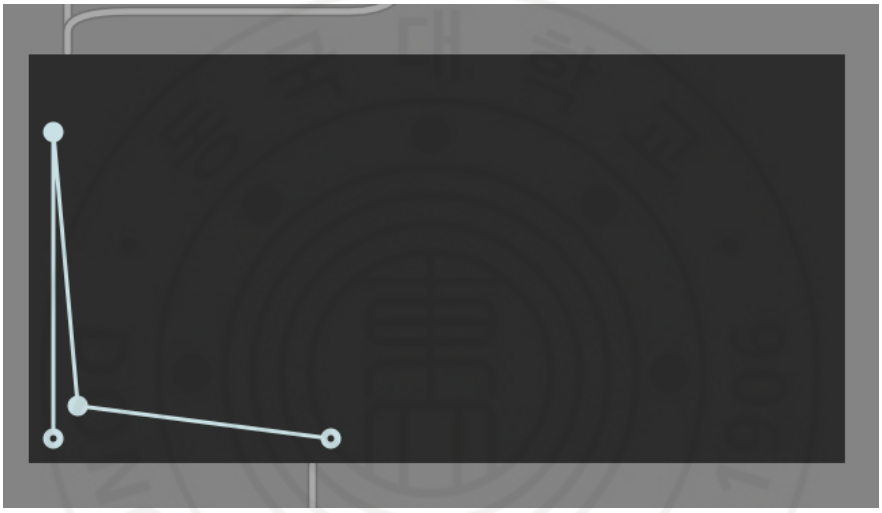
---

[11] Max for Live는 Cycling74와 Ableton에 의해 개발된 소프트웨어이다. 멀티미디어 아트 툴로 주로 이용되어지는 Max (또는 Max/MSP)를 Ableton의 DAW 소프트웨어인 Live 안에서 사용할 수 있도록 만들어져 있는 프로그래밍 개발 환경이다.

[12] Intelligent Dance Music은 전자음악의 하위 장르이다.

[13] The Karplus-Strong Algorithm . (n.d.) https://ccrma.stanford.edu/~jos/pasp/Karplus_Strong_Algorithm.html.

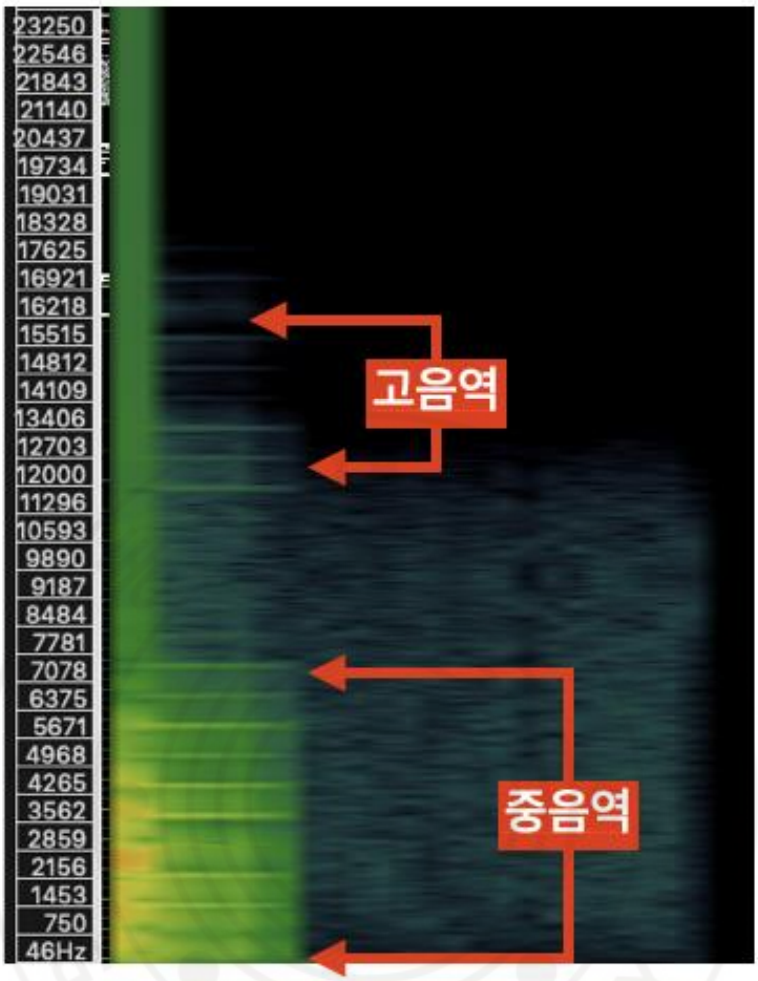
Karplus-Strong 알고리즘은 미국 Stanford 대학의 Alex Strong과 Kevin Karplus에 의해 연구된 음원 합성이다. 현을 통겨 연주(plucked string)하는 형태의 음원을 구현하기 위한 합성기법으로 사용되었다. 음색을 합성하기 위한 소스로 화이트 노이즈를 사용하며 음의 높낮이를 딜레이의 지연 값을 통해 변경하는 특징을 가지고 있다.[그림-3]



[그림-4] FeedbackNoise 의 앰프 엔벨로프

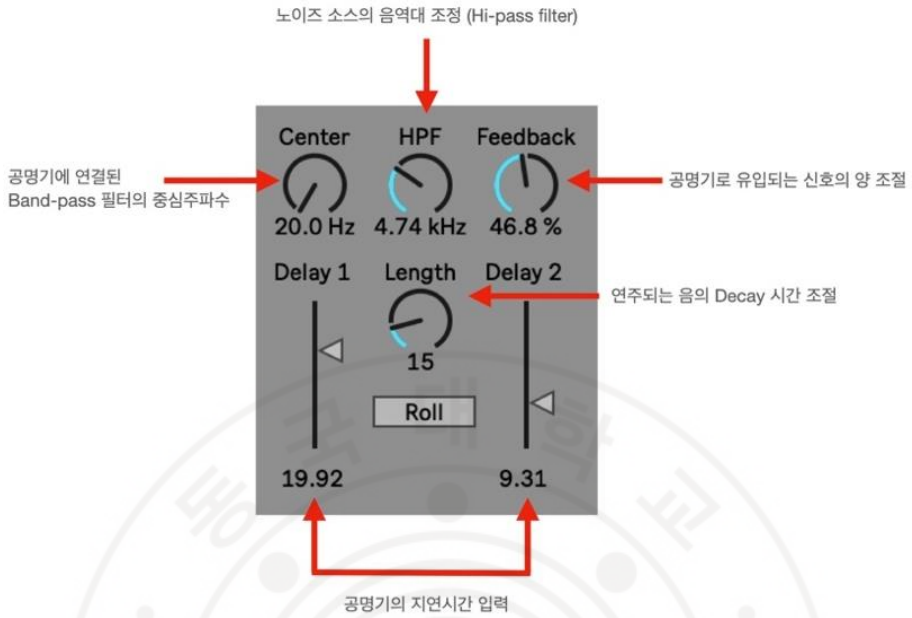
FeedbackNoise는 [그림-4]와 같이 아주 짧은 시간의 어택과 디케이를 가지는 앰프 엔벨로프(amplitude envelope)[14]를 가지고 있으며 이는 빠르고 분절된 비트 안에서의 연주를 위해 고안된 형태이다.

[14] 앰프 엔벨로프는 소리의 발생 단계를 여러 단계로 나누어 표현하는 그래프를 의미한다. 주로 어택(attack), 디케이(decay), 서스테인(sustain), 릴리즈(release)의 총 4단계로 표현된다.



[그림-5] FeedbackNoise 음원의 주파수 스펙트럼 분석

악기 FeedbackMachine의 음색은 Karplus-Strong알고리즘으로 생성되었기 때문에 주로 트랜지언트 영역에 중고음역 (1000~10000hz) 대역의 배음과 고음역대 배음이 두드러지는 것을 확인할 수 있었다.[그림-5] 하지만 신디사이저를 개발하면서 음원 합성 과정에 알고리즘이 가지고 있는 구조 외에도 필터 등 다양한 변형을 삽입함으로써 실제 음색 자체는 IDM 장르의 음악에서 많이 쓰이는 날카롭고 공격적인 음색을 합성할 수 있었다.



[그림-6] FeedbackNoise 의 사용자 인터페이스

FeedbackNoise 악기는 사용자 인터페이스의 노브와 슬라이더를 이용하여 생성되는 음색의 변화를 다양하게 줄 수 있다. 화이트 노이즈에 연결된 하이패스 필터를 이용하여 노이즈 소스의 음색 변경을 할 수 있으며 공명기(resonator)를 만드는 두 개의 딜레이 라인의 지연시간을 변경하여 타격음의 높낮이를 조절 할 수 있다. 공명기에는 band-reject 필터[15]가 연결되어 있는데 이것의 중심 주파수를 사용자 인터페이스에 위치한 Center 노브를 이용하여 변경함으로써 최종 음색의 형태를 바꿀 수 있다.[그림-6]

[15] 필터의 중심 주파수 주변부를 제외한 나머지 음역의 신호를 내보내는 필터

단일 악기만으로는 복잡한 리듬 시퀀스를 표현할 수 없었기 때문에 복수의 샘플 또는 악기를 한 트랙에서 편집하게 도와주는 Ableton Live[16]의 Drum Rack이란 악기에서 각 미디 노트마다 FeedbackNoise 신디사이저를 하나씩 삽입하여 총 16개의 다른 형태의 음원을 디자인하여 사용하게 되었다.

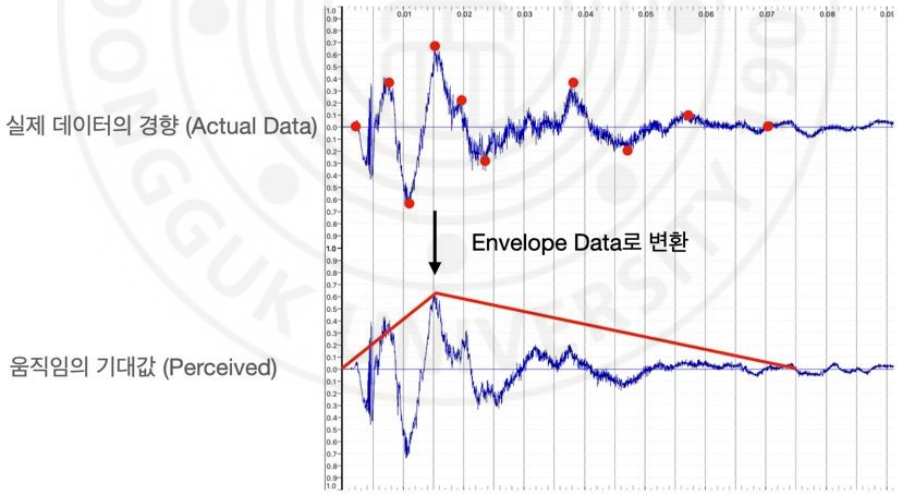
동국대학교 DONGGUK UNIVERSITY 1906

[16] Ableton Live는 독일의 음악 소프트웨어 회사 Ableton이 개발한 음악 작곡 소프트웨어이다.

2. 오디오 프로그램 디자인

1) 오디오 신호 분석 소프트웨어 3BandAudio 이펙트

본 연구에서는 음악의 리듬과 영상의 움직임이 연동되는 인터랙션의 구현에 대하여 연구하였다. 프로그램 개발 초기단계에서 드럼 연주의 오디오 신호가 가진 앰프값을 분석한 뒤 바로 영상에 적용하였으나 이 경우 연주되는 음이 가진 엔벨로프의 움직임을 직관적으로 나타내기보다 오디오 신호를 구성하고 있는 파형의 마루와 골을 모두 담고 있어 실제 값이 불규칙하게 나타나는 것을 확인할 수 있었다.

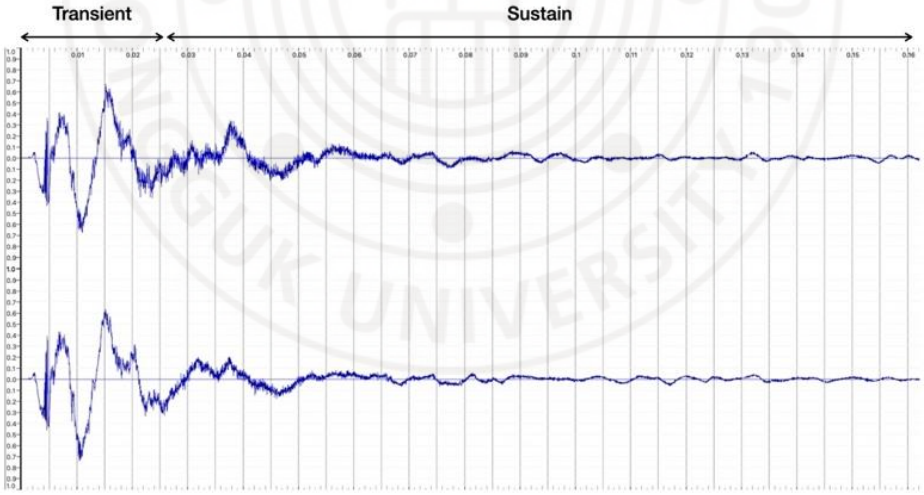


[그림-7] 실제 관측되는 데이터와 기대값의 정리

실제 오디오 데이터의 반영이라는 측면에서는 적절한 해석 없는 직접적인 데이터의 매핑이 의미가 있을 수 있다. 하지만 영상과의 인터랙션 구현에서

이러한 매핑 방식을 사용하게 되면 영상의 움직임이 직관적이지 않아 원하는 인터랙션의 기대값과 불일치하는 것을 확인할 수 있었다. [그림-7]

따라서 단순 앰프값을 추출해서 적용하는 것이 아니라 리듬의 위치(onset)를 추출해내어 데이터로 변환하는 방식이 필요했으며 이를 위해 오디오 파형에서 트랜지언트를 찾는 알고리즘을 적용하는 것이 적합하다는 결론에 다다랐다.[그림-8] 더불어 스테레오 오디오 신호에 담긴 드럼의 복잡한 패턴을 분석하기 위해서는 스테레오 오디오 신호에서 저음/중음/고음으로 신호를 분리하여 킥 드럼, 스네어, 하이햇 등을 따로 측정할 수 있어야한다. 킥 드럼이나 스네어를 서로 분리하기 위해서는 매우 좁은 음역만을 필터링해야 했기 때문에 각 음역 별로 밴드 패스(band-pass) 필터[17]를 사용하는 것이 적합한 것을 알 수 있었다.



[그림-8] 드럼 파형의 트랜지언트 / 서스테인 구분

[17] 주파수 대역폭에서 지정한 중심 주파수를 포함하는 영역의 신호만 통과시키는 필터

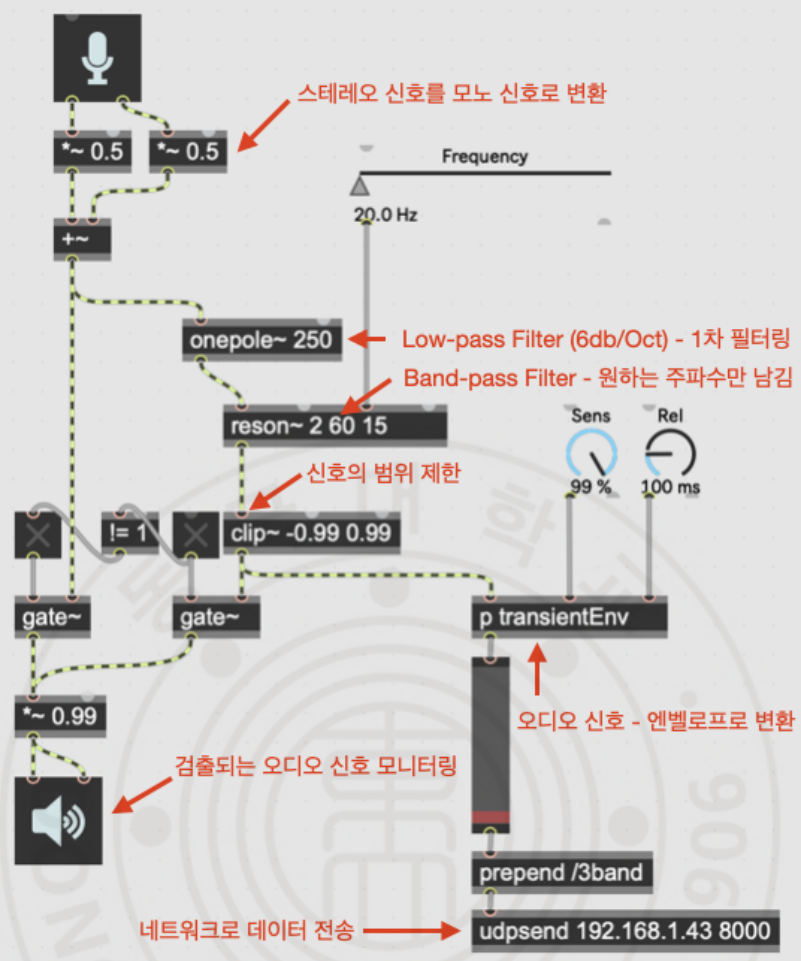
이렇게 분리된 오디오 신호에서 각 음역대의 트랜지언트의 위치를 찾아낸 후 원하는 길이의 릴리즈를 가지는 엔벨로프 신호를 생성하기 위해 노이즈 게이트(noise gate)[18]및 컴프레서(compressor)[19]에서 볼 수 있는 레벨 디텍터(level detector)[20] 알고리즘을 이용하여 구현한 프로그램인 3BandAudio 이펙터를 디자인하게 되었다.

동국대학교 DONGGUK UNIVERSITY 1906

[18] 노이즈 게이트란 음향 다이나믹 프로세서 중 하나로, 일정 레벨 이하의 신호는 밖으로 내보내지 않고 게인을 -INF에 가깝게 감쇄시켜 불필요한 잡음이나 신호의 출력을 차단하는 형태의 이펙트를 가리킨다.

[19] 컴프레서란 음향 다이나믹 프로세서 중 하나로, 일정 레벨을 넘는 신호를 지정한 비율로 감쇄시키는 형태의 이펙트를 가리킨다.

[20] 신호의 크기를 검출하는 기능



[그림-9] 트랜지언트 신호 검출 프로그램

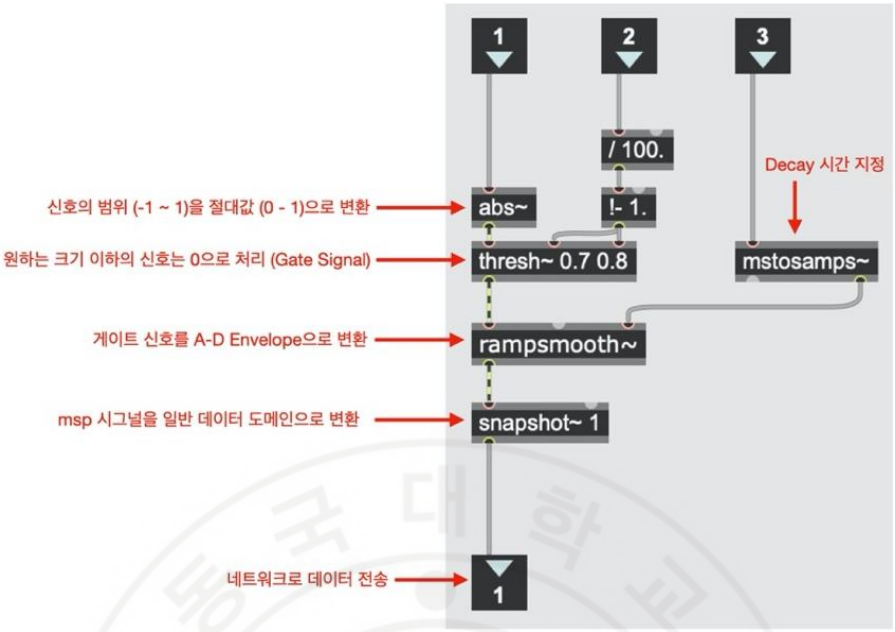
Ableton Live에서 Max for Live로 입력받는 스테레오 오디오 신호를 분석하기 위해서는 먼저 스테레오 신호를 +~ 오브젝트를 통해 모노 채널의 오디오로 변환하는 과정이 필요하다. 모노 채널로 변환하는 과정에서 음량이 증가하게 되는데, 증가되는 레벨의 변화량을 상쇄하기 위해 L/R채널의 amp

값[21]을 0.5로 변경한다.[22] 이렇게 변환된 신호에서 특정 음역의 데이터만을 검출하기 위해 입력 신호를 6db/oct 의 슬로프를 가지는 onepole~ 오브젝트의 로우 패스 필터를 통해 일차적으로 필터링한 후 밴드 패스 필터인 reson~ 오브젝트를 이용하여 원하는 주파수만 남겨놓을 수 있도록 만들어 준다. clip~ 오브젝트는 입력받는 신호를 지정한 범위 (-0.9 ~ 0.9) 이내의 값만을 가지도록 만들기 위해 연결된 것이다. clip~을 사용하면 음색에 크고 작은 왜곡을 가져올 수 있지만 시그널 데이터가 지정한 범위를 넘어가는 크기를 가지지 않게 하기 위해 사용되었다. 이렇게 필터링된 오디오 신호는 트랜지언트를 검출하여 엔벨로프 데이터로 바꿔주는 서브패치[23] [transientEnv]로 전달되게 된다.[그림-9]

[21] Max for Live에서는 오디오 시그널의 범위를 0과 1 사이의 소수형 데이터로 처리한다.

[22] MSP Panning Tutorial 2: Stereo panning . (n.d.). https://docs.cycling74.com/max8/tutorials/13_panningchapter02.

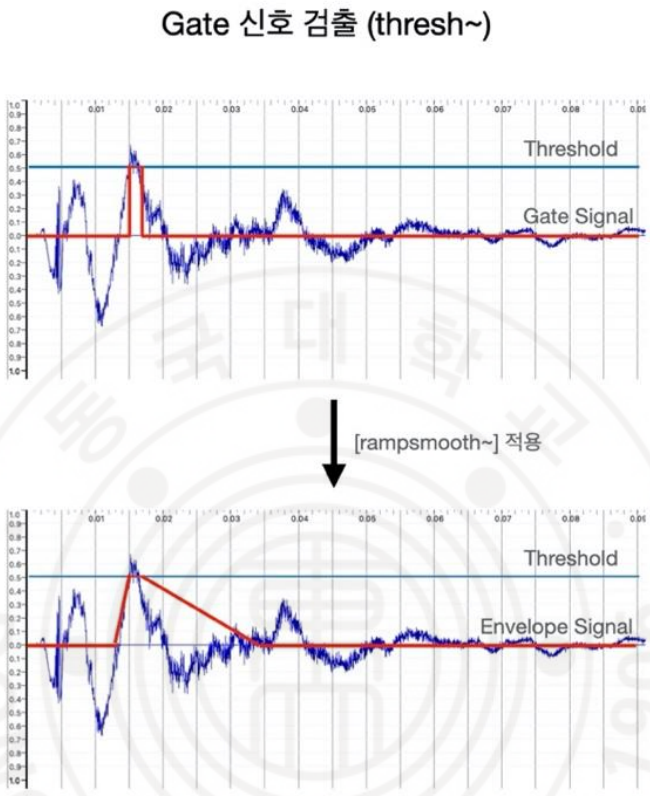
[23] 서브패치란 Max/MSP 또는 Max for Live 환경에서 프로그램 내부에 또 다른 패치를 작성하고 압축할 수 있게 도와주는 기능 및 오브젝트를 뜻한다.



[그림-10] 서브패치 [transientEnv]의 내부

[그림-10]은 서브패치 transientEnv가 오디오 신호에서 트랜지언트를 검출하고 엔벨로프 데이터로 변환하는 과정을 보여준다. 서브패치로 입력된 오디오 시그널은 -1과 1 사이의 데이터 범위를 가진다. 이 데이터의 값을 절대값으로 변환하는 abs~ 오브젝트를 사용하게 되면 양극(bipolar)의 범위를 가지는 오디오 시그널을 단극 신호 (unipolar signal)로 변환할 수 있게 된다. 이렇게 단극 신호로 변경된 오디오 신호의 크기가 특정 레벨을 넘는지 확인하는 thresh~ 오브젝트를 사용하여 오디오 시그널을 구성하고 있는 샘

플 중 지정한 threshold를 넘는 샘플을 찾아내어 게이트 신호(gate signal)[24] 의 형태로 변경한다.[25][그림-11]



[그림-11] Detector 알고리즘의 신호 검출 과정

rampsmooth~오브젝트는 입력받은 게이트 신호의 ON/OFF 과정에 지연 시간을 더하여 게이트 신호를 엔벨로프로 변환한다. rampsmooth~에 입력

[24] 게이트 신호란 펄스 (사각파)의 형태를 가지는 디지털 신호를 뜻하는데 이는 특정한 시간 동안 일어나는 이벤트의 길이를 표현할 때 사용되는 신호를 뜻한다.

[25] [thresh~]의 출력은 오직 두가지 상태 중 하나만을 가지도록 만들어져 있다. 지정한 역치(Threshold)를 넘지 않았을 때는 0, 넘은 경우를 1의 시그널로 표현하여 게이트 신호를 생성한다.

되는 지연시간은 오디오신호가 처리되는 속도, 즉 샘플레이트(sample rate)[26]에 기반한 길이로 입력되어야 하며 이를 구하는 식은 다음과 같다.

$$T = \lfloor \frac{S.R}{1000} \rfloor$$

오디오 지연 시간 T는 샘플레이트 S.R을 1초 (1000ms)로 나눈 값을 내림한 값이다. 즉, 시스템이 44.1khz의 샘플레이트로 작동하는 경우 1ms는 약 44 sample의 길이를 가진다고 말할 수 있다. 해당 수식을 rampsmooth~에 적용하기 위해서는 expr오브젝트 등을 이용하여 직접 수식을 입력하는 방법도 있으나 이 프로그램에서는 mstosamps~[27]라는 오브젝트를 사용했다. 이렇게 처리된 신호를 snapshot~ 오브젝트를 이용하여 OSC[28]에 담을 수 있는 데이터로 변환한다. 여기서 처리된 데이터는 udpsend오브젝트를 통해 영상 제어 데이터와 함께 외부 네트워크로 전송되게 된다.

---

[26] 샘플레이트란 디지털 오디오 인터페이스가 아날로그 오디오 신호를 디지털 신호로 옮기는 샘플링 속도를 의미한다.

[27] [mstosamps~]는 현재 작동하고 있는 오디오 시스템의 상태를 호출하고 샘플 길이를 계산하는 함수를 내장하고 있기 때문에, 수식 표현 없이 이를 이용하여 실시간으로 지연시간을 계산해 낼 수 있다.

[28] OpenSoundControl. OSC는 CNMAT에서 개발한 콘텐츠 포맷으로, 신디사이저 및 컴퓨터, 다른 멀티미디어 장비들을 공연이나 쇼에서 네트워크를 통해 제어할 수 있도록 만들어진 프로토콜을 뜻한다.

2) AudioGrabber - FFT 분석과 Jitter 매트릭스를 이용한 악곡 분석

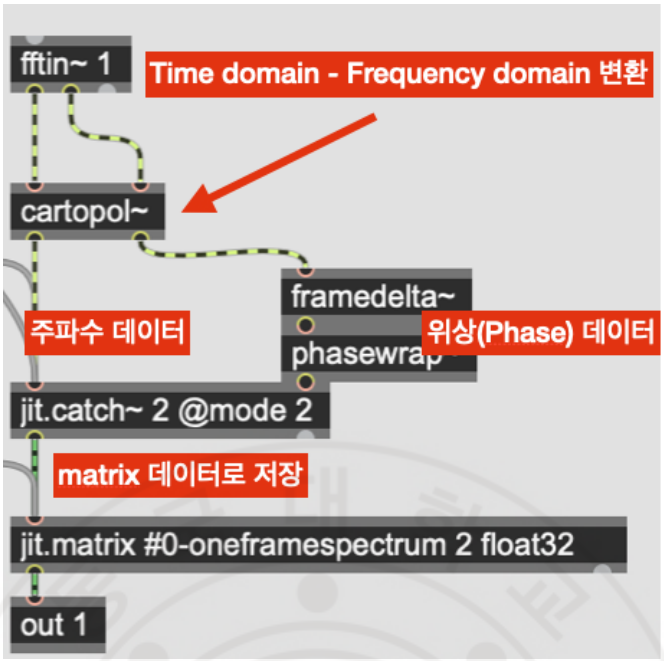
AudioGrabber는 음악의 전체 주파수 스펙트럼을 FFT[29] 데이터로 저장하고 전송하는 프로그램으로 Max for Live를 이용하여 제작되었다. 음악의 악상이 변화하는 추세를 데이터화하는 방법으로 Jean-Francois Charles가 제시한 기법을 참고하였다.[30]

푸리에 변환 과정은 일반적으로 시간 영역의 신호를 주파수 영역의 값으로 변환하는 과정으로 설명된다. 이를 수학적으로 표현할 경우, 데카르트 좌표계(Cartesian coordinate system)의 데이터를 극좌표계(polar coordinate system)으로 옮기는 과정이라고도 설명할 수 있다.[그림-12]

---

[29] Fast Fourier Transform, 고속 푸리에 변환은 이산적인 시간 안의 집합에서 시간 영역(Time domain)의 신호를 이용하여 주파수 영역 (Frequency domain)의 데이터로 변환, 실제 음원(complex tone)의 Sine 성분을 분해하는 과정을 뜻한다.

[30] Jean-Francois는 FFT분석을 마친 시그널 데이터를 Jitter를 이용하여 그래픽 데이터로 변환한 뒤, Jitter 오브젝트들을 이용해 스펙트럼 데이터에 그래픽 프로세싱을 가하고 다시 IFT(Inverse Fourier Transform)를 통해 다시 오디오 신호로 재합성하는 기법을 제시하였다. (Charles, J. n.d. A Tutorial on Spectral Sound Processing Using Max/MSP and Jitter. Computer Music Journal, 32, pp. 87-102.)



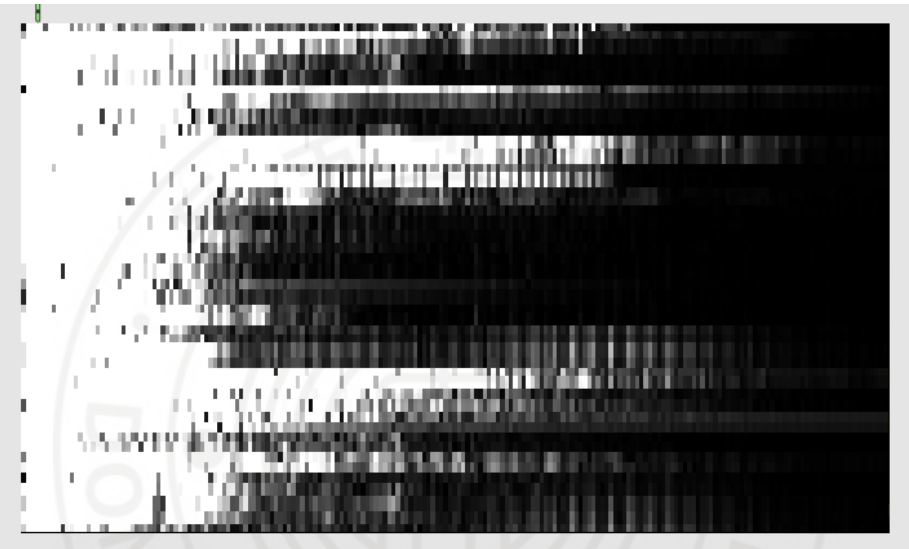
[그림-12] AudioGrabber 에서 사용된 pfft 패치

Max for Live에서 고속 푸리에 변환을 위한 연산[31]은 MSP 시그널 데이터에 `cartopol~` (cartesian to polar)라는 오브젝트를 사용하여 시그널에서 주파수(frequency) 데이터와 정위(phase) 데이터를 추출하는 과정으로 간략화 된다. 이렇게 FFT를 통해 변환된 데이터는 `jit.catch~`오브젝트에서 float32 데이터로 변환되어 1평면 매트릭스[32]에 2차원 배열로 저장된다. Max for Live는 OSC 외에도 매트릭스 데이터를 직접 UDP 프로토콜을 통

[31] Max는 푸리에 변환(Fourier transform)과 관련된 프로세스를 실시간으로 처리할 수 있는 환경을 지원하는데 FFT전용 포맷인 pfft의 형태로 서브패치를 디자인하여 사용할 수 있도록 하고 있다.

[32] Jitter에서 매트릭스란 수학적인 의미보다는 2차원 배열에 가까운 데이터 컨테이너의 의미로써 쓰인다.

해 전송하는 기능을 지원한다. 오디오 클라이언트에 설치된 AudioGrabber 프로그램은 Live의 마스터 버스 채널의 오디오 데이터를 최소한의 지연시간 안에 무선으로 전송하기 위해 256x256 사이즈의 이미지로 다운 샘플링하여 그래픽 서버[33]로 전송하게 된다.



[그림-13] 그래픽 서버로 전송되는 matrix 데이터 (Max)

매트릭스 데이터를 픽셀 이미지로 변환해서 보여주는 jit.pwindow오브젝트를 통해 오디오 스펙트럼 이미지를 관찰하게 되면 실제 전송되는 오디오 데이터가 [그림-13]처럼 흑백의 주파수 스펙트럼 이미지로 변환되어 전송되는 것을 알 수 있다. 이렇게 픽셀 데이터로 변환된 주파수 스펙트럼 데이터들은 화면에서 0에 가까운 값(오디오 신호 없음)일 수록 어두운색으로

[33] 본문의 그래픽 서버란 별도로 설치된 영상 제어 시스템이 설치된 컴퓨터를 뜻한다.

표시되고 1에 가까운 값(오디오 신호가 있음)일 수록 밝은 톤으로 표시된다. 즉 악상이 진행될수록 음원의 스펙트럼을 담은 매트릭스에서 밝은 픽셀들의 분포가 증가하고 악상의 내용이 적어질수록 어두운 영역이 커지는 것으로 해석할 수 있다. 수신받은 이미지에서 각 픽셀이 밝은 색을 가질수록 1에 가까워지는 것에 착안하여 입력받은 매트릭스를 전체 데이터가 가진 평균(mean), 최솟값(minimum), 최댓값(maximum)을 출력하는 오브젝트인 jit.3m으로 보내 픽셀의 평균값을 구하여 음악의 재생 상황을 수치화하였다. 이미지 픽셀의 평균값 데이터가 커지는 경우를 악상의 변화나 악기 편성이 증가하는 것으로 볼 수 있고 음악에서 연주되는 악상의 내용이 적어질수록 평균값이 낮아진다고 판단할 수 있게 된다. 이를 통해 작품의 시각적 요소들인 파티클 시스템(particle system), 포인트 클라우드(point cloud) 및 시각화의 주 요소인 화면 중앙의 물체에 악상의 변화에 따른 시각적 변화를 매핑한 영상을 만들 수 있게 되었다.[그림-14]

[그림-14] 악상의 변화에 따른 영상 변화

3. 그래픽 서버

그래픽 서버는 Max[34]의 그래픽 라이브러리인 Jitter를 통해 디자인되었다. 서버 내부의 기능은 각종 생성 예술에서 사용되는 그래픽 요소인 파티클 시스템(particle system), 포인트 클라우드(point cloud) 및 noise displacement 테크닉을 접합시킨 추상적인 구체를 생성하는 메인 시스템과 화면에 출력되는 픽셀 및 스크린의 효과를 조정하는 렌더링 엔진으로 나뉘며, 렌더링 및 그래픽 보정을 위한 라이브러리로는 Jitter 내에서 GLSL 3[35]를 사용할 수 있게 해주는 GL3라이브러리를 사용하였다.

실제 음악 내에서 전달되는 오디오 트랜지언트의 반응과 동기화된 애니메이션을 구현하기 위한 영상 프로토타이핑 과정에서 동시에 많은 수의 연산을 요구하거나 병목 현상이 일어나는 부분들이 존재하였기 때문에 생각보다 많은 시스템 자원을 효율적으로 소모할 수 있는 형태의 프로그램 디자인이 필요하였다. Jitter에서는 그래픽 알고리즘의 구현 뿐만 아니라 `qlim`, `jit.qball`과 같은 프로세스 최적화 오브젝트를 가지고 있다. 그래픽 프로세싱에 필요한 컴퓨팅 자원을 CPU뿐만 아니라 GPU에서 할당할 수 있는 기

---

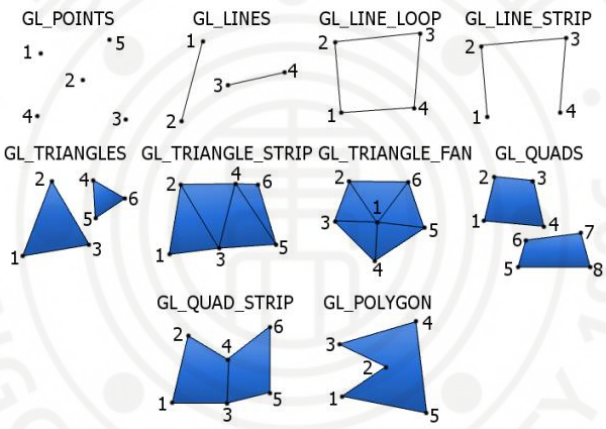
[34] Max는 Cycling '74에서 개발한 인터랙티브 미디어를 구현하기 위한 소프트웨어이다. 데이터의 연산과 처리 및 프로그래밍이 가능한 Max와 음향 시그널 데이터 처리가 가능한 MSP, 그리고 real-time video 및 2D/3D 그래픽을 다루는 Jitter로 나누어져 있다.

[35] OpenGL Shading Language. GLSL은 고수준의 쉐이딩 언어로 C언어 문법을 기본으로 작성된다. ARB 어셈블리어를 사용하지 않고도 그래픽 파이프라인에 직접적인 제어가 가능하도록 만들어진 언어이다.

능을 쉽게 구현할 수 있게 도와주기 때문에 Jitter를 그래픽 서버 구성 언어로 선택하게 되었다.

1) jit.bfg 오브젝트를 활용한 noise displacement 기법[36]

OpenGL 그래픽 라이브러리에서 3D 공간의 모든 물체의 형상은 버텍스(vertex)[37]의 조합으로 정의 내려진다. 두 버텍스가 가진 좌표를 잇게 되면 선(line)이 되고, 3개 이상의 버텍스가 모이면 삼각형, 다수의 버텍스가 모여 다각형 형태인 폴리곤(polygon)이 된다. 컴퓨터 그래픽스에서는 이러한 요소를 2차원 기초 요소(primitives)라고 부른다.[그림-15]



[그림-15] OpenGL 의 Geometric Primitives[38]

---

[36] noise displacement란 노이즈 알고리즘을 이용하여 물체를 이루고 있는 버텍스의 좌표를 비트는 기법을 뜻한다.

[37] 버텍스는 3D 공간을 이루고 있는 가장 작은 단위의 데이터 세트로, 입체 공간의 X/Y/Z 좌푯값을 가지는 벡터를 뜻한다.

[38] Tutorial 2 - 2D Primitives . (n.d.). https://taskercode.wordpress.com/tutorials/opengl-primer/tutorial-2-2d-primitives/.

Jitter는 기본적으로 가장 단순한 형태의 기하학 요소들을 3D 공간에서 쉽게 사용할 수 있도록 다양한 함수들을 지원한다. 대표적인 오브젝트로는 jit.gl.gridshape[39]와 jit.gen[40]이 있다. 이러한 오브젝트들은 가상 공간에 sphere, torus, cylinder, opencyclinder, cube, opencube, plane, circle와 같은 기본 형상을 쉽게 구현할 수 있도록 도와줄 뿐만 아니라 형상을 이루는 버텍스 데이터에 직접 접근하여 새로운 형상을 생성할 수 있게 해준다. 이러한 물체의 형상들은 Jitter에서 각 도형의 버텍스가 가진 좌푯값을 담고 있는 행렬(array)의 묶음으로 표현될 수 있으며, jit.gl.gridshape와 같은 오브젝트에서 버텍스 데이터는 jit.matrix[41]라는 오브젝트를 통해 외부로 출력이 가능하다. jit.gen은 jit.matrix와 관련된 연산을 CPU를 통해 처리할 수 있도록 만들어져 있다. 동일한 프로세스를 GPU를 통해 처리할 수 있게 해주는 오브젝트인 jit.gl.pix 또한 존재한다. 본 작품에서는 형상의 프로토타이핑을

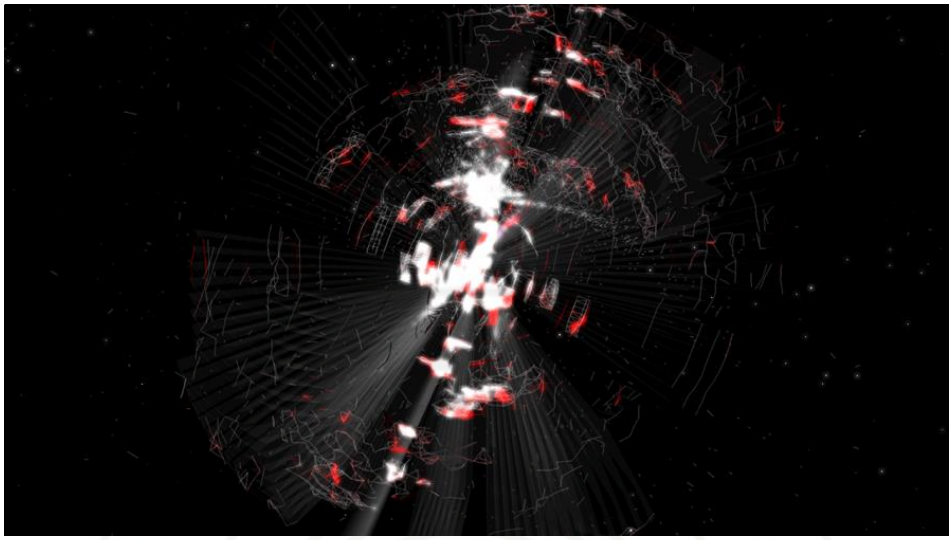
---

39) [jit.gl.gridshape]는 물체의 형태를 이루고 있는 버텍스(vertex)의 좌표를 담고 있는 매트릭스와 이를 면의 형태로 접합시켜 화면에 렌더링하는 메쉬(mesh) 및 텍스쳐(texture), 빛의 음영을 표현하기 위한 노멀 맵(normal map)등을 포함하고 있는 오브젝트 및 컨테이너이다.

40) [jit.gen]은 Max의 내장 런타임인 Gen의 Jitter 버전이다. Gen은 버텍스의 움직임을 만들기 위한 벡터 연산 혹은 OpenGL 쉐이더 프로그래밍에서 쓰이는 함수들을 텍스트 기반의 환경 및 Max와 같은 노드 기반의 프로그래밍 환경을 통해 사용할 수 있도록 고안된 오브젝트 및 개발 환경을 뜻한다.

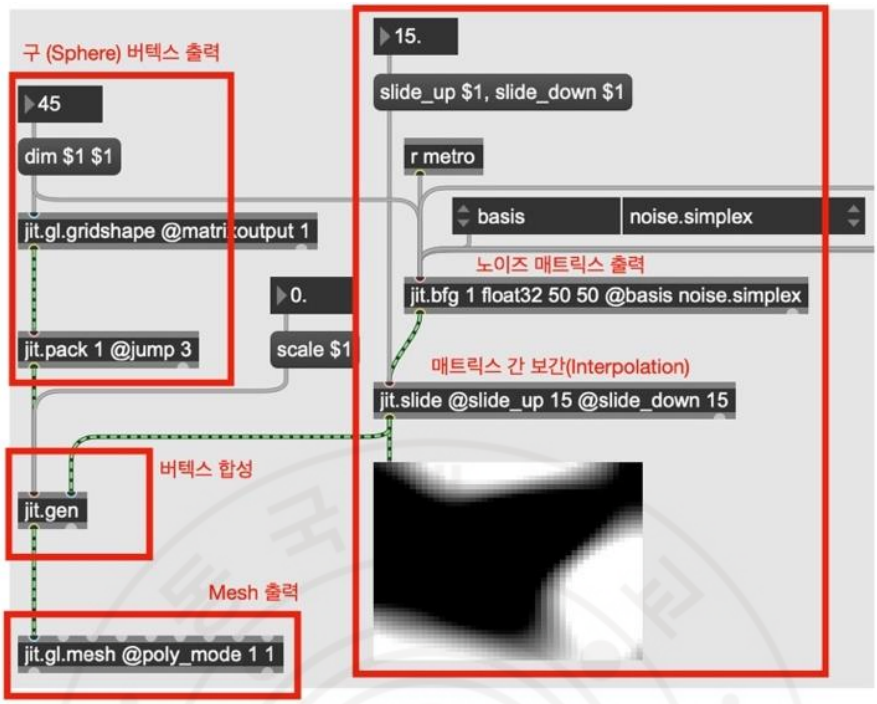
41) 가상 공간의 물체를 이루는 버텍스의 X/Y/Z좌표와 같은 복잡한 데이터를 Jitter에서 처리하기 위해 만들어진 오브젝트로 단순한 숫자 데이터가 아닌 다수의 많은 데이터를 동시에 처리할 수 있도록 디자인되었다.

jit.gen으로 진행한 후 시스템 적용에서는 최적화를 위해 jit.gl.pix로 바꾸었다.



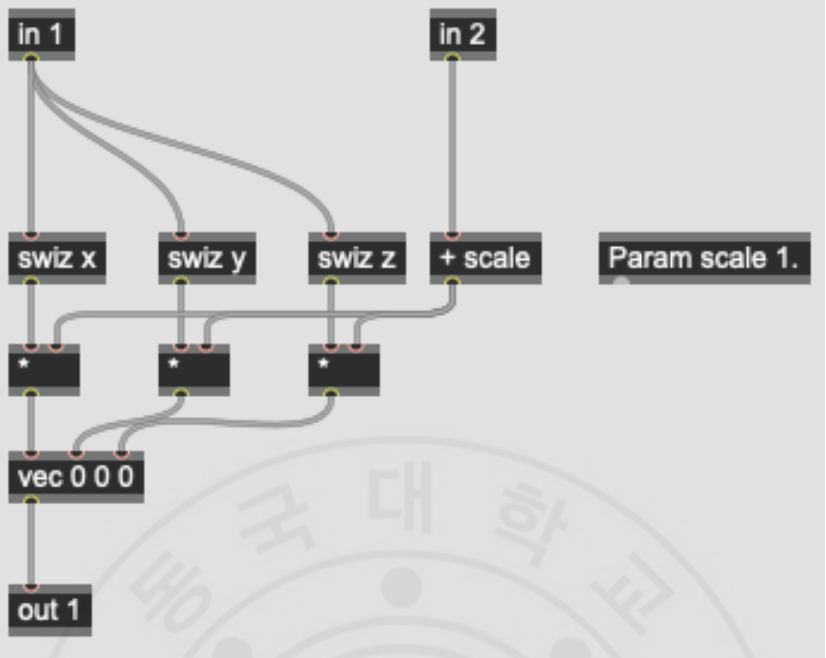
[그림-16] 작품 Event Horizon II 의 한 장면

본 작품의 주 요소인 추상적인 기하학 구상을 보여주는 [그림-16] 속 중앙의 구체는 noise displacement 기법과 다양한 쉐이더 프로그램을 통해 구현되었다. 그 중 가장 중심이 되는 기법인 noise displacement 알고리즘의 적용 과정은 다음과 같다.



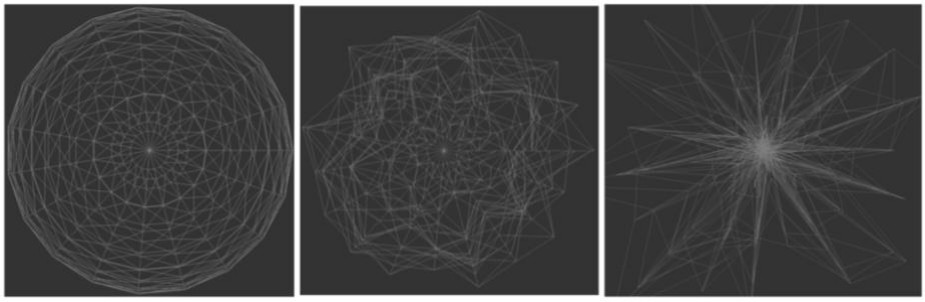
[그림-17] Noise displacement 의 알고리즘 구조도

jit.gl.gridshape는 처음 가상 공간에 인스턴싱(instancing) 됨과 동시에 화면에 출력이 가능하나, 실제 내부 데이터를 직접 수정하는 것은 불가능하다. attribute 옵션인 @matrixoutput 플래그를 활성화하면 jit.gl.gridshape가 담고 있는 매트릭스 데이터를 출력 가능하게 만들어준다. 오브젝트가 출력하는 매트릭스 데이터 중 버텍스 정보를 담고 있는 첫 0/1/2번 (X/Y/Z) 평면만을 처리하기 위해서는 jit.pack오브젝트를 통해 매트릭스를 필터링하는 과정을 거쳐야 한다. 이렇게 필터링된 데이터는 jit.gen을 통해 jit.bfg 오브젝트가 생성한 노이즈 데이터 매트릭스와 합성되고 그 결과값이 jit.gl.mesh오브젝트에 입력되어 가상 공간에 출력된다. [그림-17]



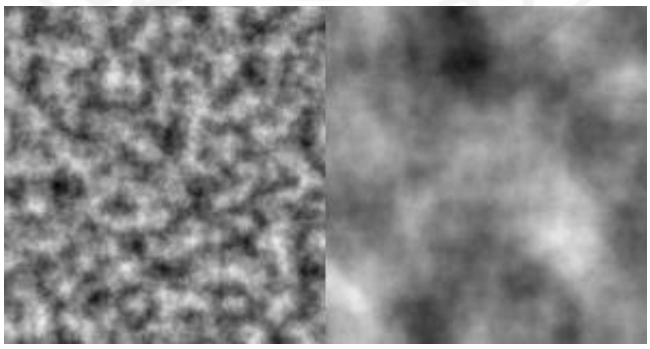
[그림-18] Jit.gen 코드 구조

jit.gl.gridshape오브젝트에서 출력된 버텍스의 X/Y/Z 좌표 데이터는 [그림-18]의 jit.gen patcher 속에서 각각 노이즈 데이터와 곱해져 새로운 벡터 값을 가지게 된다. 우측 [in 2]에서 입력되는 데이터는 무작위 값을 가지며 전체적인 매트릭스 값의 경향은 노이즈 알고리즘에 따라 달라진다.



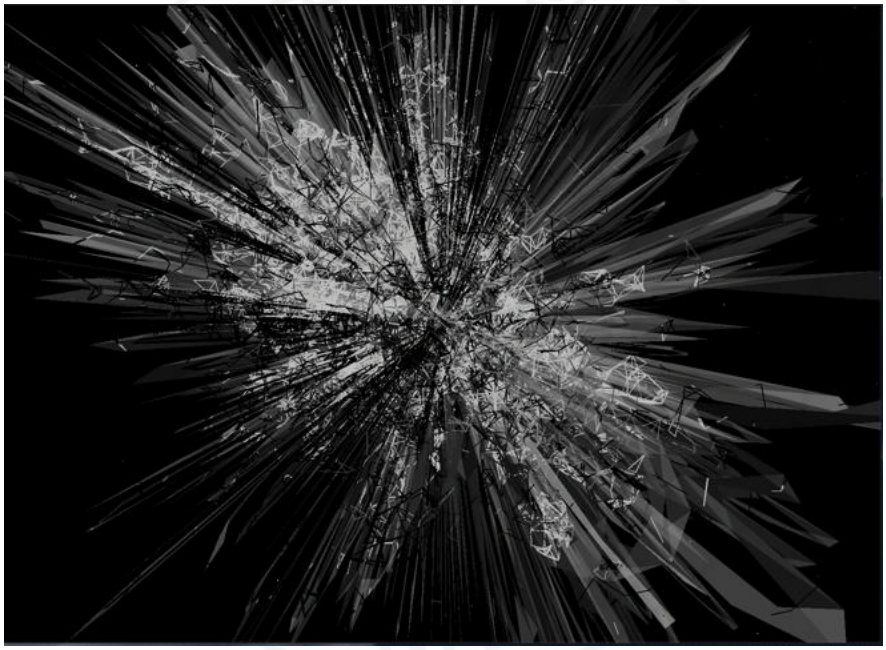
[그림-19] 구(Sphere)와 simplex 노이즈의 합성 과정

위의 알고리즘을 이용해 합성된 결과물의 변화 과정은 [그림-19]와 같다. [그림-19]에 사용된 노이즈 알고리즘은 simplex 노이즈로, Perlin 노이즈 알고리즘에서 컴퓨팅 연산의 부하를 해소시킨 형태의 알고리즘이다. 원을 구성하고 있는 버텍스의 좌표가 노이즈 알고리즘으로 생성한 난수 데이터와 결합하면서 원의 중심으로 수축하거나 밖으로 뻗어나가는 형태의 형상으로 바뀌게 된다.



[그림-20] 고주파 노이즈(좌) 와 저주파 노이즈(우)

구의 형상에 가장 큰 변화를 주는 것은 노이즈의 간격을 조정하는 것으로, jit.bfg 오브젝트에서는 scale 이라는 파라미터로 접근 가능하며 일반적인 컴퓨터 그래픽스에서는 이를 노이즈의 주파수(frequency)를 증가시킨다고 말한다. [그림-20]처럼 노이즈의 주파수가 증가하여 생성되는 난수의 간격이 넓어질수록 3D 공간상의 구가 가진 버텍스의 값 차이가 좀 더 두드러지게 나타나 조금 더 날카로운 형태로 변화하게 된다.



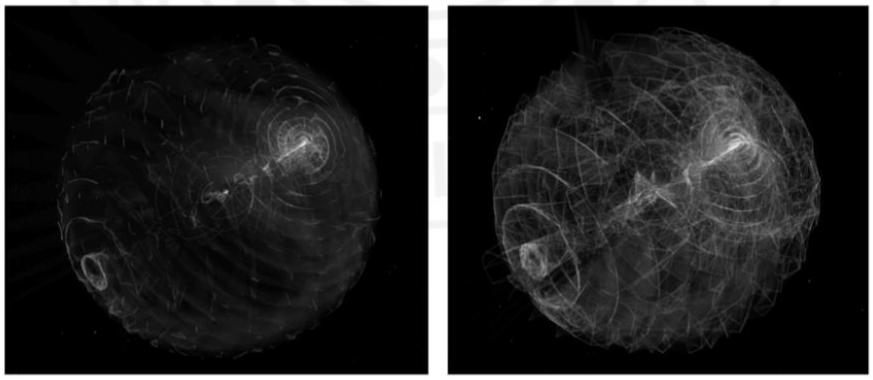
[그림-21] 최종 완성된 구체의 모습

[그림-21]은 다양한 노이즈 알고리즘을 적용한 복수의 메쉬를 이어붙여 만든 결과물로 jit.bfg 의 checker, simplex, distorted 노이즈 알고리즘을 구체에 합성한 형상을 겹쳐서 만들어낸 것이다. 이외에도 다음에 설명할 쉐

이더(shader) 프로그램을 이용하여 메쉬의 버텍스가 서로 연결되지 않고 끊어진 듯한 모습을 구현했다.

2) OpenGL Shader 프로그램 (GLSL)

OpenGL은 그래픽 렌더링 파이프라인의 작동 및 GPU 계산에 직접적인 제어를 가능하게 해주는 고수준 언어를 탑재하고 있는데 이를 쉐이더(shader) 또는 GLSL(OpenGL Shader Language)라고 부른다. 쉐이더는 C언어 기반의 문법으로 작성할 수 있다. GLSL은 현재 OpenGL을 관리하는 크로노스 그룹(Kronos Group)에 의해 여러 차례 업데이트되어 버전 4에 이른다. Jitter에서 현재 지원하는 버전은 GL 3.30 버전이며 아래에 설명될 쉐이더 프로그램 또한 330 버전에 맞춰 작성되었다.



[그림-22] 지오메트리 쉐이더를 이용한 형상 변화

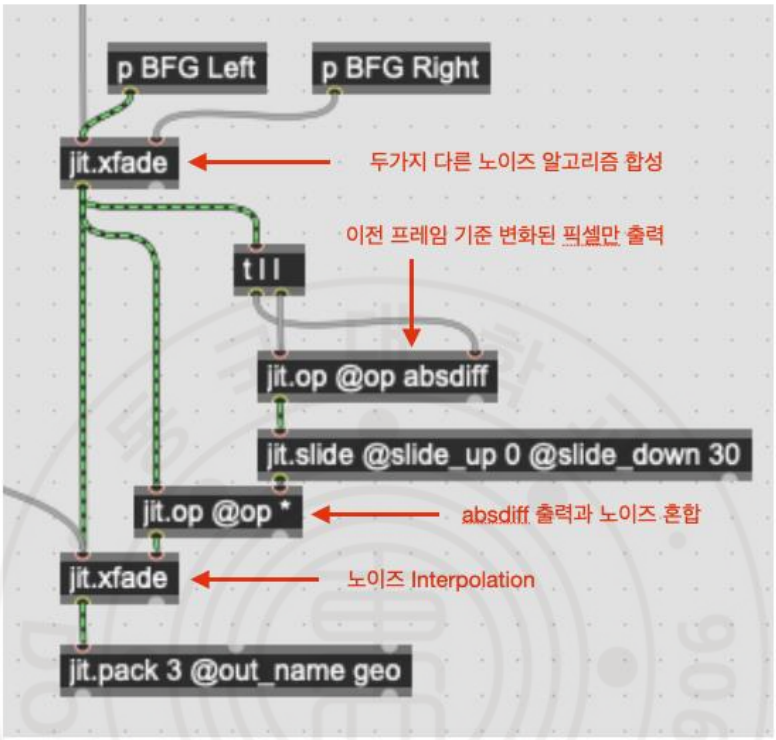
작품에서 실시간으로 형상이 끊임없이 변하는 추상적인 물체의 표면적을 디자인하기 위해서 3개의 메쉬에 다양한 노이즈를 서로 교차 시켜 구체에 합성하였다. 중심 모델을 구성하고 있는 메쉬의 경우 버텍스 간의 거리를 계산, 노이즈로 인해 버텍스 간 거리가 일정 거리 이상으로 멀어지게 되면 연결을 끊는 형태의 지오메트리 쉐이더를 추가하여 최종 형상을 조정하였다.[그림-22]

```
out jit_PerVertex {
  flat vec4 color;
  flat vec4 closeParticlePosition[arrayLength];
} jit_out;

void main() {
  int iterations;
  iterations = dim;

  gl_Position = modelViewProjectionMatrix * vec4(position,2);
  jit_out.color = color;

  // initialize close particles position to this particle position
  for (int i=0; i<arrayLength; i++)
  {
    jit_out.closeParticlePosition[i] = modelViewProjectionMatrix * vec4(position,2);
  }

  int counter = 0;
  for (int i=gl_VertexID+1; i<iterations; i++)
  {
    vec3 otherPosition;
    float dist;

    otherPosition = texelFetch(posBuffer, i).xyz;
    dist = length(position - otherPosition);
    // if(i % 2 == 0){
    //  otherPosition = texelFetch(posBuffer, i).xyz;
    //  dist = length(position - otherPosition);
    // }else{
    //  break;
    // }

    if (dist < distThreshold && dist > 0.0)
    {
      jit_out.closeParticlePosition[counter] = modelViewProjectionMatrix * vec4(otherPosition,2);
      counter++;
      if (counter >= arrayLength)
      {
        break;
      }
    }
```

[그림-23] 프로그램에 사용된 Geometry Shader 의 한 부분

[그림-23]은 실제 프로그램에 사용된 OpenGL 지오메트리 쉐이더의 한 부분으로, 각 버텍스가 가진 XYZ 좌표간의 거리(dist)가 지정된 범위(distThreshold) 이내에 들어오는 경우 각 버텍스를 선으로 잇게 만들어주는 프로그램이다. 전체 과정 중 입력되는 버텍스 데이터를 지정된 길이(이 경우 입력받는 매트릭스의 크기) 동안 검색하여 두 점 간의 거리를 찾아내는 부분이며, 루프가 지정된 범위를 벗어나 GPU에 불필요한 부하를 가하지 않도록 멈출 수 있게 만들어져 있다.

3) 이미지 프로세싱을 활용한 노이즈 데이터 변조



[그림-24] 노이즈 데이터의 변조 과정

noise displacement 기법에서는 노이즈의 값이 구체의 형상 변화에 직접적인 영향을 주게 되는 구조의 프로그램을 디자인하는 과정을 설명하였다. 해당 시스템에서 노이즈의 위치(origin)이나 크기(scale)등을 조작하면 실제 구체를 이루고 있는 버텍스의 벡터 연산으로 이어져 형상 변화뿐만 아니라 애니메이션 또한 입력이 가능해짐을 알 수 있었다. 추가적으로 작품에 사용된 노이즈에 이미지 프로세싱 기법을 적용하여 새로운 텍스쳐를 실시간으로 생성 및 변조하여 생성해낸 텍스쳐가 형상의 변화에 영향을 주는 방식으로

물체의 움직임을 만들어냈다. 이 과정에서 주로 사용된 효과는 두 픽셀값을 서로 전환하는 crossfade 효과인 jit.xfade오브젝트와 이전 프레임과 현재 프레임 사이의 변화량만을 출력하는 오브젝트인 absdiff[42] 이다. 이들을 사용하여 노이즈 텍스쳐를 변조하였다. [그림-24]

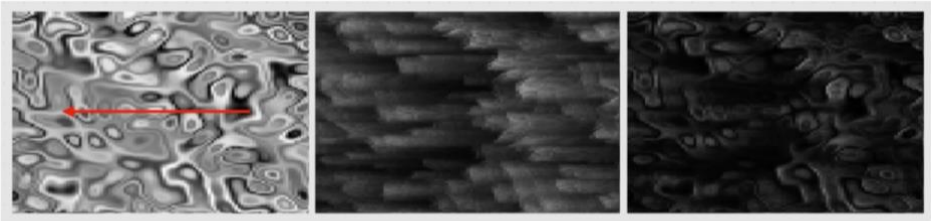
jit.op오브젝트는 매트릭스 안의 데이터에 지정한 연산자(operator)를 사용하여 사칙연산 및 비트연산 등을 버텍스가 담긴 매트릭스에 적용할 수 있게 하는 오브젝트이다. 그 중 연산자 absdiff는 OpenCV[43]같은 픽셀 데이터를 이용한 모션 트래킹에 쓰이는 오퍼레이터인데 본 작품에서는 버텍스 데이터에 합성될 노이즈 데이터의 변화량을 추적하는데 사용되었다.

jit.bfg오브젝트로 생성된 노이즈는 offset 메시지를 통해 프로그램이 켜진 동안 x축으로 계속 진행하게 된다. [그림-25]의 가운데 그림은 absdiff로 처리된 텍스쳐로써, 진행 중 변화가 있다고 표시되는 구간만을 출력한 상태를 나타낸다. 이후 다시 크로스페이드를 통해 입력 노이즈와 합성된 결과물 (그림-25의 우측)이 실제 구체에 매핑되는 텍스쳐이다.

---

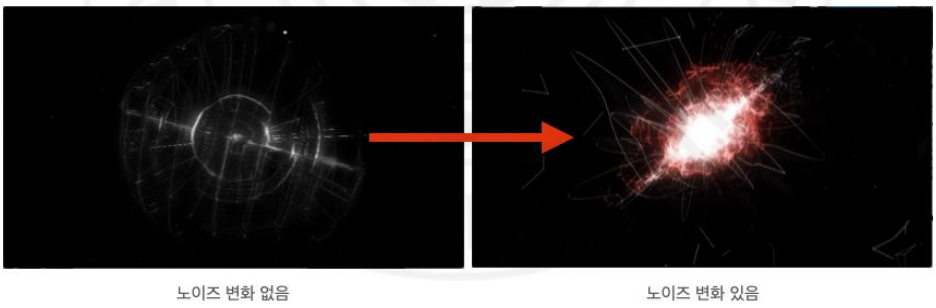
[42] 연속된 픽셀 데이터에서 이전 프레임과 현재 프레임의 픽셀의 변화값을 절대량으로 출력하는 오퍼레이터. 컴퓨터 이미지에서 정적인 배경과 움직임이 있는 물체를 구별할 때 사용하는 알고리즘이다.

[43] OpenCV(Open Source Computer Vision)은 실시간 컴퓨터 비전을 목적으로 한 프로그래밍 라이브러리이며, 실시간 이미지 프로세싱에 중점을 둔 라이브러리이다.



[그림-25] 노이즈 데이터의 이미지 프로세싱 과정

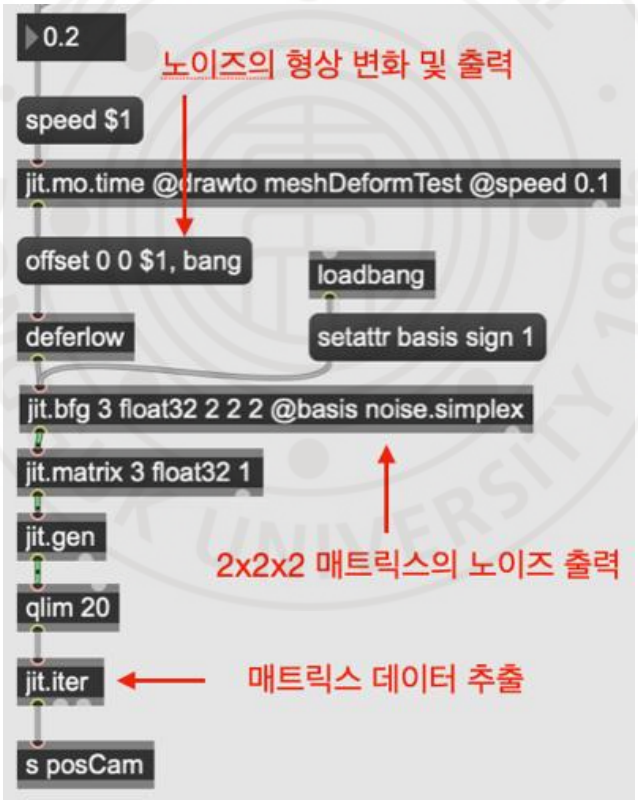
실제 작품에 사용된 노이즈 알고리즘은 사용자의 의도에 맞게 임의의 엔벨로프를 이용하여 노이즈의 상태를 급격히 바꾸도록 프로그래밍되어 있다. 빠른 움직임이 일어날 경우 absdiff 는 조금 더 많은 픽셀을 출력하게 되고 그 결과는 [그림-26]처럼 원형의 구체에서 수축되거나 회전되는 듯한 모습으로 변화하게 되는 걸 볼 수 있다.



[그림-26] 노이즈 조작에 따른 구체의 형상 변화

4) 노이즈 알고리즘을 이용한 실시간 시각점 변환

작곡한 음악이 가진 빠른 리듬의 역동성을 강조하기 위해서는 단순히 구체의 형상을 변경하는 것 뿐만 아니라 추가적인 효과가 필요했다. 음악의 전개 과정에 따라 역동적인 카메라워크를 만들어낼 수 있는 방법으로써 `jit.bfg`를 이용해 생성되는 난수 데이터를 카메라 좌표의 소스로 활용하는 방식을 택했다.

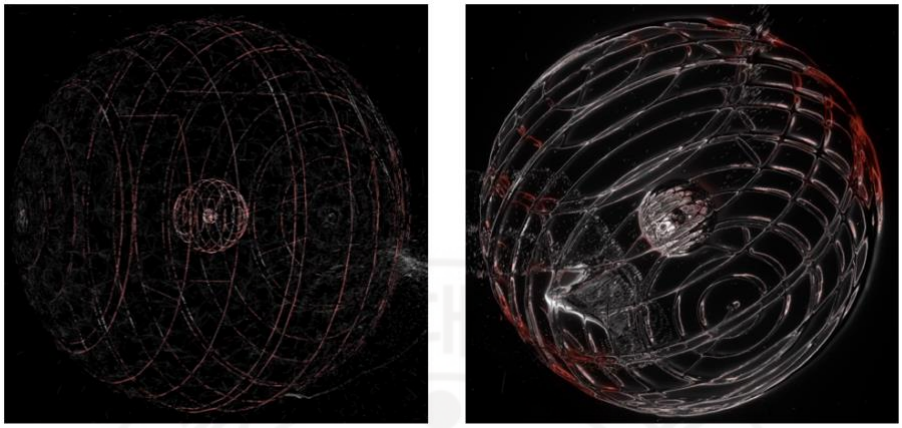


[그림-27] 카메라 좌표 생성 과정

jit.bfg 오브젝트는 개별 데이터로 나눌 경우 32비트 길이를 가지는 소수 데이터로 사용 가능하다. jit.bfg 전용 명령어인 'basis sign 1'을 이용하면 생성되는 난수의 범위를 +1 ~ -1 사이로 지정할 수 있었고 이를 통해 중심점 (0,0,0)의 가장자리를 3차원 방향으로 회전하는 X/Y/Z 좌표를 생성할 수 있었다. 이렇게 생성된 데이터는 매트릭스의 데이터를 맥스의 데이터 영역으로 출력하는 jit.iter 오브젝트를 통하여 list로 출력이 된다. 이 과정에서 불필요하게 빠른 카메라 시각점의 전환이 만들어낼 시스템의 병목현상을 없애기 위하여 qlim오브젝트를 통해 20ms 간격으로 카메라로 전달되도록 만들었다. [그림-27]

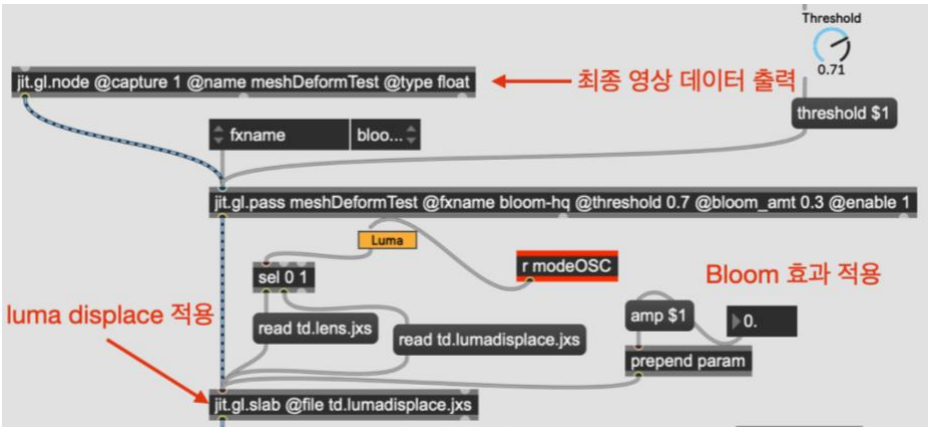
OpenGL에서는 카메라의 시각점을 간단히 고정할 수 있는 명령어인 locklook이 존재한다. 해당 명령어를 Jitter의 카메라 오브젝트인 jit.gl.camera에 입력함으로써 중심점을 바라본 채로 카메라가 회전하게 만들어 실제 오브젝트에 적용한 noise displacement 효과가 조금 더 역동적으로 적용되고 구체의 모든 면을 조망할 수 있도록 설정하였다.

5) 렌더링 프로세싱



[그림-28] 이펙트 미 적용(좌), luma displace 적용 (우)

Jitter에서는 다양한 쉐이더(Shader) 효과를 최종적으로 렌더링되는 영상에 사용할 수 있도록 지원한다. 본 연구에서는 해당 이펙트들을 사용하여 렌더링되는 픽셀의 밝기(luma)를 기준으로 광택 또는 액체처럼 보이게 만드는 입체적인 효과를 영상에 적용할 수 있었다.[그림-28]

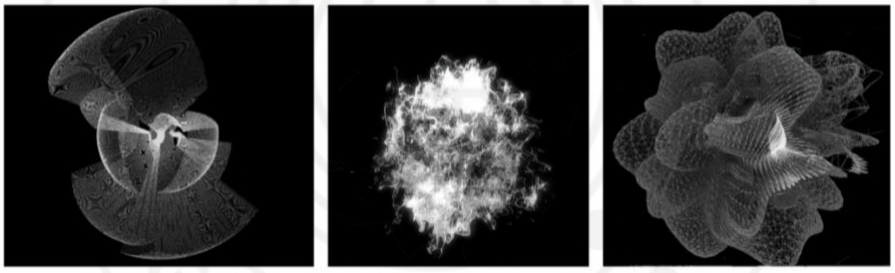


[그림-29] Max 의 post-processing 과정

[그림-29]와 같이 jit.gl.node오브젝트를 이용하면 현재 Jitter를 통해 만들어지는 결과물을 텍스쳐로 받아와서 처리할 수 있게 만들어 준다. jit.gl.node 오브젝트로 전송받은 최종 출력물을 jit.gl.pass오브젝트로 보내 bloom 이펙트를 주었으며, jit.gl.slab오브젝트를 이용하여 Jitter 내장 쉐이더 프로그램중 하나인 luma displace 효과를 적용시켰다.

6) 시스템 최적화 및 영상의 구체화

160BPM[44]의 빠른 템포에서 연주되는 드럼의 트랜지언트에 반응하는 영상을 만들기 위해서 3만 개 이상의 버텍스의 변형 과정과 쉐이더 프로그램을 최소한의 지연 속도로 재생할 수 있게 시스템을 최적화하는 과정이 필요했다. 더불어 시스템의 안정성 및 자원 배분의 효율성을 위해 영상 시스템과 음악 시스템을 분리하였다. 또한 무선으로 데이터를 주고받아야 했기 때문에 영상 시스템에 버텍스 개수를 제한하거나 프로세싱 속도를 조정하는 등의 다양한 설정값을 테스트해보면서 최종 결과물을 만들어냈다.[그림-30]

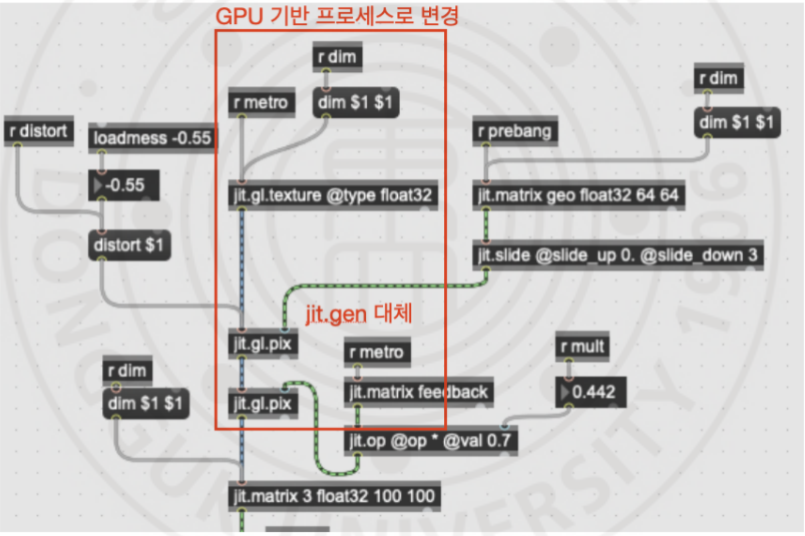


[그림-30] 구체의 모핑(morphing) 형상 테스트 과정

첫 아이디어 구현 단계인 [그림-30] 좌측에서 발전되어 다양한 형상을 테스트하는 과정에서 약 10,000개의 버텍스를 이용한 구체를 사용했을 때 시스템에 무리가 가지 않고 초당 60프레임의 프레임률을 보여주는 것을 확인

44) Beats Per Minute. 분당 비트의 개수, 음악의 템포를 나타낼 때 사용되는 단위

할 수 있었다. 이렇게 디자인한 구체가 가진 버텍스의 좌표가 빠르게 움직이는 인터랙션을 매핑하자 jit.gl.gridshape 오브젝트에서 프로세스 과부하가 걸리면서 프레임률이 초당 40회 이하로 감소하는 것을 확인할 수 있었다. 실제 렌더링 되는 속도보다 더 빠르게 많은 데이터가 전송되었기 때문에 단순히 프레임 수가 감소하는 것 뿐만 아니라 구체의 움직임 자체에 지연이 발생하여 원래의 결과물이 보여주던 빠른 움직임을 만들 수 없게 되었기 때문에 이를 해결할 방법이 필요했다.

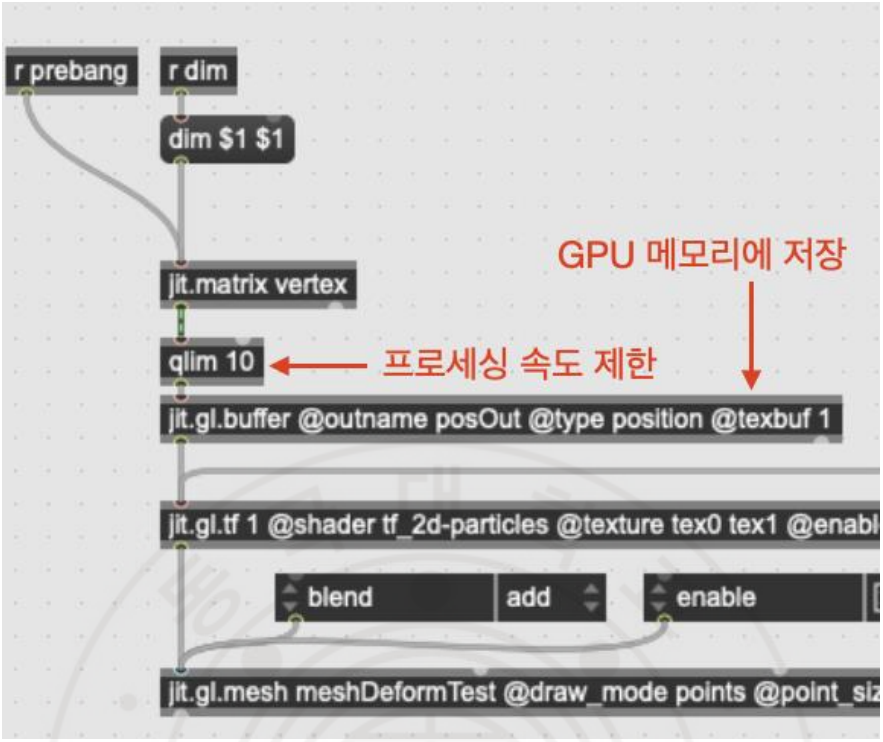


[그림-31] 실제 작품에 사용된 코드

프로세스의 지연을 가져온 jit.gl.gridshape는 구체의 버텍스 데이터 외에도 텍스쳐와 같은 다양한 데이터를 매트릭스로 출력하기 때문에 실제 작품에 사용된 코드에서는 구체의 버텍스 데이터 외에 불필요한 데이터는 제외하고 버텍스 데이터만 생성할 수 있는 오브젝트로 교체해야 했다. [그림

-31]에서 볼 수 있듯 구체를 이루는 버텍스의 생성을 jit.gl.gridshape가 아닌 jit.gl.pix로 변경하였다. 이를 통해 구체의 데이터를 보내고 처리하는 과정을 GPU에서 실행할 수 있도록 변경하여 프로세스가 지연되는 것을 막을 수 있었다.

영상 시스템 내에서 처리되는 대부분의 프로세싱은 동시에 많은 데이터의 계산을 요구한다. Jitter 프로그램의 제어뿐만 아니라 OS및 다양한 애플리케이션의 프로세싱을 담당하는 CPU로는 원활한 처리가 불가능 했기 때문에 대신 GPU의 연산 유닛들을 사용하는 것이 효율적임을 확인할 수 있었고 이를 통해 약 20FPS 정도의 성능 향상 (1920x1080해상도 기준)을 가져와 초당 60프레임의 속도를 유지할 수 있었다.



[그림-32] 프로세싱 속도 제한 및 최적화 과정

Jitter에서는 다양한 방식으로 프로세싱 우선 순위 및 프로세싱 속도 제한을 패치에 적용할 수 있도록 한다. [그림-32]처럼 `qlim`을 사용하면 지정된 간격(ms 단위)안에서만 데이터가 다음 오브젝트로 전송하도록 제한할 수 있게 된다. 이러한 방법으로 카메라 시각점 이동(50ms 제한) 및 OSC 데이터 입력(20ms 제한) 등 매 프레임이 그려지는 속도보다 느리게 전송해도 문제가 없는 부분들은 불필요하게 프로세싱 자원을 소모하지 않도록 속도를 제

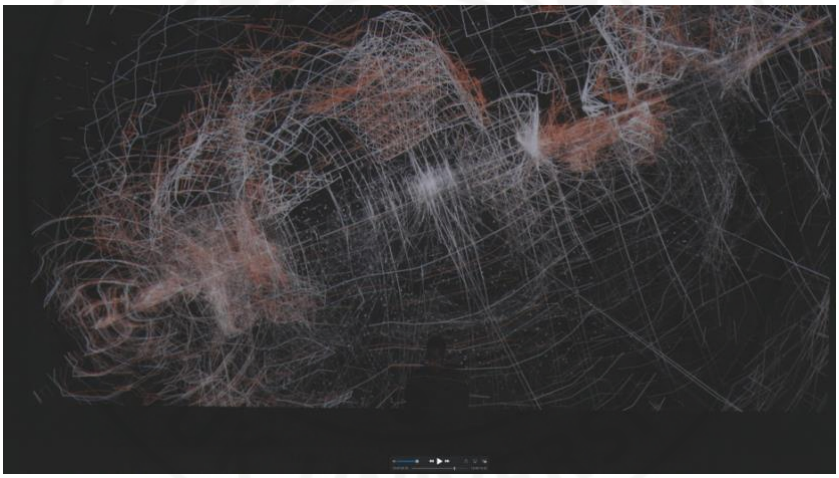
한하여 시스템의 안정성을 높였다. 이후 남은 CPU / GPU 자원들은 FSAA[45], bloom 등 OpenGL에서 지원하는 포스트 프로세싱 이펙트들을 사용하는데 이전해주어 결과물의 품질을 향상시킬 수 있었다.

[45] Full-Screen Anti-Aliasing (FSAA)는 실제 렌더링되는 화면보다 더 높은 해상도의 이미지를 GPU에서 렌더링한 뒤 다운스케일링하여 대각선 등에서 생기는 그래픽 깨짐 현상을 줄이는 효과이다.

III. 연구 기술의 작품 적용

연구된 오디오비주얼 인터랙션 시스템을 이용한 멀티미디어 작품 <Event Horizon II>는 2021년 11월 13일 동국대학교 이해랑 예술극장에서 진행된 한국멀티미디어음악학회의 'SEEING SOUND, LISTENING IMAGE XVIII' 공연에서 초연되었다.

1. 작품 소개



[그림-33] 실제 공연 이미지

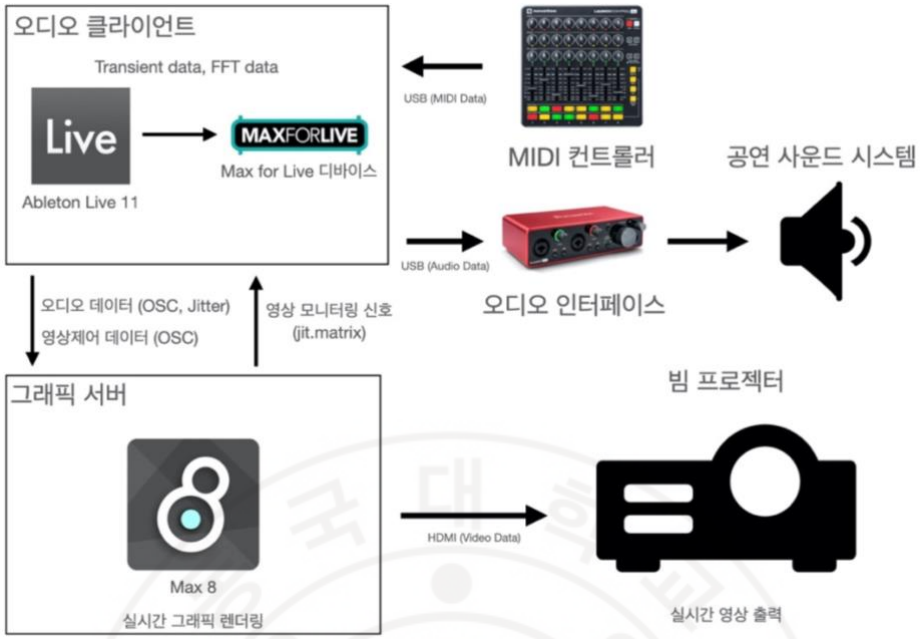
작품 <Event Horizon II>는 실시간으로 음악의 리듬에서 트랜지언트를 추출하여 이를 통해 영상이 제어되는 멀티미디어 작품이다. 노이즈, 글리치 기반의 사운드와 변칙적인 비트에 반응하는 추상적인 형상의 변화 과정을 포착한 본 작품은 악곡과 영상의 인터랙션을 위해 영상과 음악이 동시에 작곡

되었으며 음악의 구성 또한 실제 영상의 변화되는 폭을 반영하여 구성되었다. 따라서 제작한 Jitter 프로그램이 생성해내는 영상은 직접적인 오디오의 변화 뿐만 아니라 간접적인 음악의 맥락 또한 반영한 결과이며 음악의 악상과 진행에 따라 변화되는 것이 특징이다.[그림-33]

2. 작품 구성

1) 무대 및 시스템 구성

본 작품을 구현하는 시스템의 구성도는 [그림-34]와 같다. 프로젝터를 제외한 두 부분은 오디오 클라이언트(audio client), 그래픽 서버(graphic server)로 나누어진다. 오디오 클라이언트는 Ableton Live를 통해 음악을 오디오 인터페이스로 전송하는 부분과 연주되는 곡의 트랜지언트(transient)를 분석하여 그래픽 서버로 보내는 Max for Live 디바이스 및 음원 전체의 스펙트럼 데이터를 FFT로 분석한 다음 이를 이미지로 변환하여 TCP/IP를 통해 전송하는 Max for Live 디바이스를 포함한 시스템을 뜻한다. '그래픽 서버'는 별도의 컴퓨터에 설치되어 WIFI를 통해 원격으로 오디오 클라이언트 컴퓨터에서 데이터를 입력받아 실제 3차원 공간상의 물체의 움직임과 특징을 변화시키고 이를 시각화하는 시스템을 가리킨다.



[그림-34] 공연 시스템 구성도

2) 음악 구성

본 작품은 크게 Intro, A, B, C, Outro 파트로 나누어지며 음악의 전체적인 구성은 <표-1>과 같다. 음악의 도입부는 사인파 신디사이저의 벨 패치로 만든 곡의 테마가 시작되면서 FeedbackNoise로 디자인된 글리치 리듬 시퀀스가 전개되기 시작한다.

A 파트는 악곡의 테마를 이루는 멜로디가 반복되면서 악상이 전개되는 과정을 담고 있다. 음악 전체의 리듬 아이디어를 담당하는 Karplus-Strong 신디사이저의 리듬 시퀀스에 드럼 키트가 함께 연주되면서 음악의 테마와 영상이 확장되는 과정을 표현한다. 이후 등장하는 신디사이저 리드가 테마를 보조하는 라인을 연주하면서 B 파트로 전환된다.

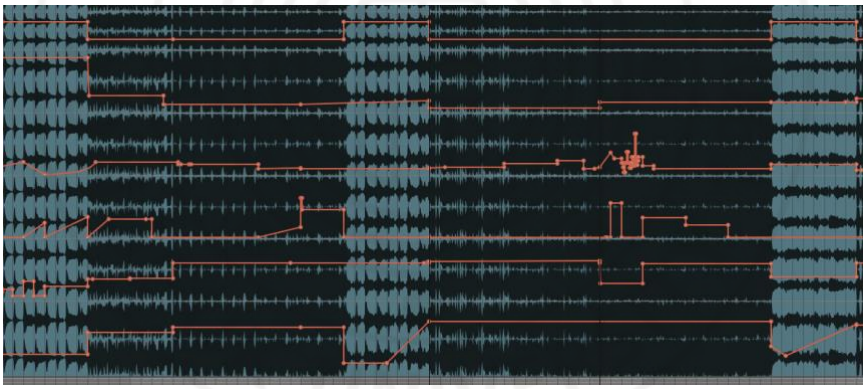
B 파트에서는 변칙적인 리듬 파트의 전개와 함께 사운드와 영상의 연결 관계를 탐색할 수 있는 구간이 진행된다. 영상 속 물체의 변화와 음악의 인터랙션을 보여주기 위한 구간으로 물체의 움직임을 음악적으로 표현한 파트로 지속음들 대신 짧게 분절되는 연주들이 나타나며 전체적으로 영상 속 이미지가 수축하는 과정을 표현한다.

C 파트는 파트 A/B 에서 보여주었던 아이디어를 기반으로 점진적으로 에너지가 강하게 상승하는 구간으로 영상의 전개에 가장 큰 변화를 가져오는 부분이다. 드론 신디사이저와 패드가 글리치 드럼을 보조하면서 음악 전체의 다이내믹이 가장 강하게 상승하는 구간으로 영상 속 물체가 확장되면서 화면 전체를 가득 채우게 되는 부분을 음향적으로 표현한다.

<표-1> 작품 구성

구성	Intro (0:00~00:36)	A (00:36~1:30)	B (1:30~4:12)	C (4:12~6:36)	Outro (6:36~7:12)
중심요소	사인 벨,글리치	노이즈 퍼커션	노이즈 퍼커션, 글리치	드론,패드	사인 벨
실시간모양변화		확장	수축	확장	
영상효과	Noise displacement	Noise displacement	Noise displacement	Luma displacement	Noise displacement

작품에 사용된 영상 시스템은 끊임없이 새로운 형상을 생성할 수 있도록 디자인되어 있었지만 음악과의 인터랙션을 강조하기 위해서는 단순한 엔벨로프 매핑과 형상의 자동 생성뿐만 아니라 실제 형상을 이루게 하는 몇 가지 설정값을 악곡의 분위기와 맥락에 맞게 변환하는 개입 없이는 작품의 완결성을 가질 수 없었다. [그림-35]는 악곡의 진행에 따라 오디오 클라이언트의 Ableton Live에서 OSC로 보내는 데이터들을 보여준다. 이 데이터들은 구체의 크기, 형상을 뒤트는 알고리즘 속 파라미터나 카메라의 회전 속도와 같은 화면을 구성하는 기초적인 설정값을 지정하였다.

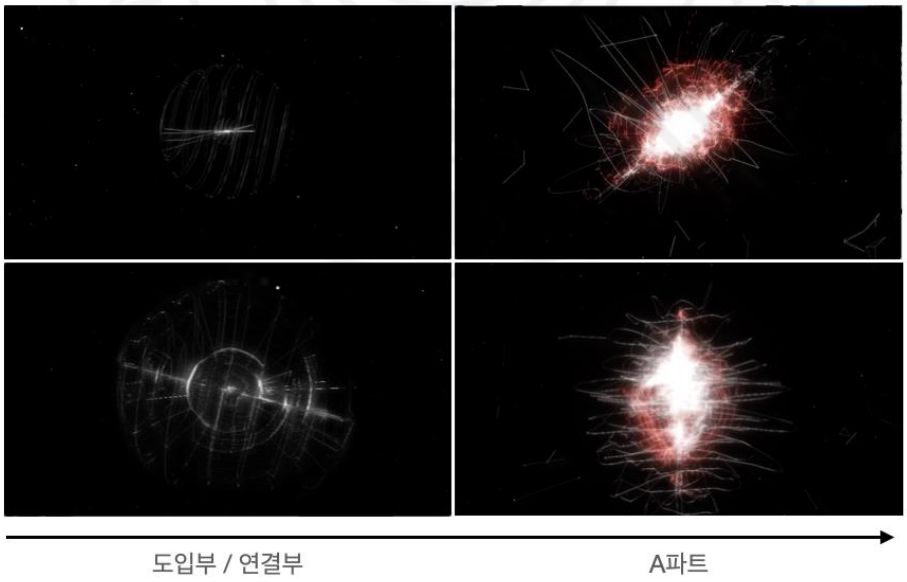


[그림-35] 오디오 클라이언트에서 디자인된 파라미터 값들

3) 영상 구성

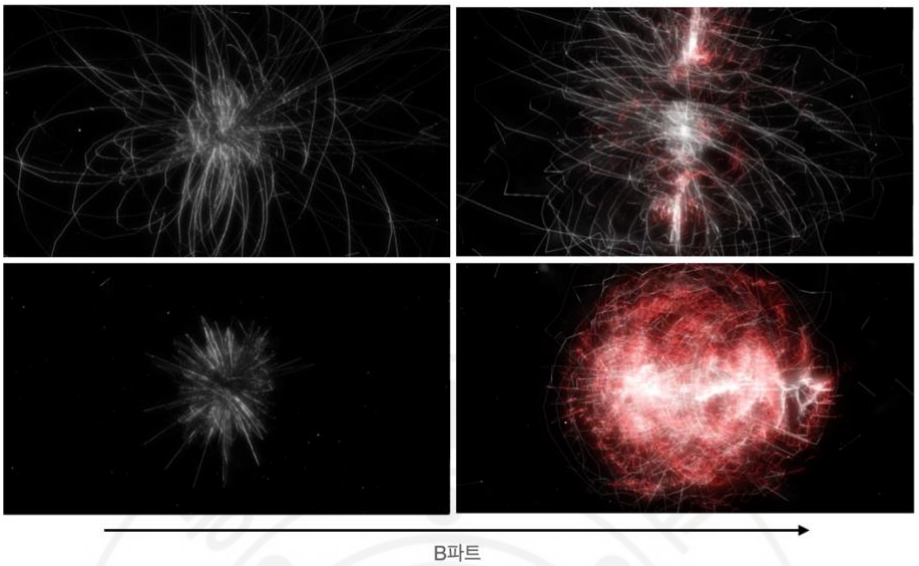
① 도입부 / A 파트

영상의 도입부 및 A 파트의 진행에서 보여주는 영상의 변화는 [그림-36]과 같다. 음악에서 리듬이 없거나 반복적인 비트가 나오지 않는 경우 화면 중앙의 물체는 구(sphere)의 형상을 유지하고 있으며 느리게 움직인다. 첫 장면에서는 구체를 이루고 있는 버텍스의 개수가 250개 이하로 제한되며 단순한 형태를 띠게 된다. 음악에서 리듬 요소가 등장하면서 점차 형상이 발전해 나가게 되며, 이후 킥 드럼이 연주되면 저음부의 트랜지언트에 따라서 물체의 중심을 이루고 있는 메쉬(mesh)가 빨간색으로 변화하며 리듬이 가진 강세를 나타내게 된다.



[그림-36] A 파트의 영상 변화

② B 파트



[그림-37] B 파트의 영상 변화

B 파트의 영상으로 음악이 진행되면서 물체의 크기와 형상의 변화를 주었다.[그림-37] 특히 B 파트에서는 다양한 물체의 변화 과정과 함께 음악이 진행되면서 구체의 형상 변화와 리듬 간의 상관관계를 알아볼 수 있는 악구를 추가하여 리듬이 빠르게 변화할수록 각 형상으로 변형되는 속도를 빠르게 바꾸어 A 파트와 대비를 이루도록 만들었다. 이후 B 파트가 발전하면서 악곡 안의 이벤트 양이 늘어나게 됨에 따라 오디오 클라이언트가 그래픽 서버로 보내는 데이터의 양이 증가하게 되고 배경을 이루고 있는 point cloud의 점들 또한 서로 연결되면서 배경화면을 가득 채우게 된다.[그림-38]

[그림-38] B 파트의 공연 사진

③ C 파트

[그림-39] C 파트의 공연 사진

C 파트는 작품 내에서 가장 강한 인상을 보여주는 파트로, drone[46] 신디사이저와 복잡한 리듬 텍스쳐가 만들어내는 음향 공간을 시각화하기 위해 만들어졌다. B 파트에 이어서 나오는 구체에는 쉐이더 프로그램인 luma displace[47]효과가 적용되면서 메쉬 전체가 액체와 같은 질감으로 바뀌어 전반부와는 다른 질감을 나타내며 음악과 마찬가지로 영상의 정보량을 늘려나가기 시작한다. 형상의 변화뿐만 아니라 영상을 조망하고 있는 카메라의 시점 또한 구체의 내부로 진입하면서 관객들은 구체가 품고 있던 복잡한 형상

---

[46] 음악에서 드론drone이란 화성 또는 단음의 효과 또는 반주를 일컫는 단어로, 작품 전체에서 긴 음을 지속적으로 연주하는 형태를 뜻한다.

[47] Jitter 라이브러리에 내장된 GLSL 쉐이더 이펙트의 이름을 뜻한다.

을 좀 더 자세하게 관찰할 수 있게 된다. 구체의 운동 속도 및 카메라의 전환 속도 또한 A파트와 다르게 조금 느리게 조정되어 전체적인 형상과 음악의 상호작용을 느낄 수 있는 부분이다.[그림-39]

④ Outro

[그림-40] Outro 의 공연 장면

C 파트가 끝나고 모든 연주가 잦아들면서 결말의 부분으로 향하게 되면 음악의 스펙트럼 데이터의 정보량이 줄어들게 되는데 이로 인해 구체의 크기도 다시 도입부와 마찬가지로 작게 변형되며, 포인트 클라우드도 배경에서 사라진다. 파티클 시스템은 emitter의 위치가 카메라 시점을 따르도록 설정되어 있기 때문에 구체의 주변부를 맴도는 형태의 애니메이션을 만들면서. 도입부의 공간으로 다시 돌아온 듯한 느낌을 만든다.

3. 연구 기술의 작품 적용 효과

본 작품은 노이즈가 음향 및 시각매체에서 사용되는 여러 다양한 기법을 적용한 멀티미디어 작품이다. 음향적 노이즈 요소를 음악에 사용하기 위해 도입된 Karplus-Strong 알고리즘 합성 기법은 본래 알고리즘이 만들어지면서 제안되었던 방식인 현악기 음원을 재현하는 용도 뿐만 아니라 전자음에 가까운 글리치(glitch) 톤의 타악기의 소스로도 효과적으로 사용될 수 있음을 본 작품을 제작하면서 확인할 수 있었다. 이를 통해 작품에서 나타내고자 했던 유기적이지만 이질적인 노이즈 타악기 시퀀스를 작곡할 수 있었다.

영상에서는 시각적 노이즈를 물체의 형상에 합성하는 기법인 noise displacement를 사용하여 지금까지 미니멀리스트 작품에서 보여지는 흑/백의 다소 차가운 질감의 영상 언어에서 벗어난 새로운 표현 방식을 발견할 수 있었다. 뿐만 아니라 Jitter에서 GLSL 쉐이더(Shader) 프로그램을 사용하여 영상 표현을 극대화하였으며 이를 통해 관객이 좀 더 작품에 몰입할 수 있도록 하였다. 리듬 정보를 검출하는 프로그램과 음악의 정보량을 이미지 데이터로 변환하는 방식을 연구하여 음악과 영상이 함께 연동되는 비주얼라이징 시스템을 제작했으며 작곡가의 의도와 실제 음악 정보가 일치되어 영상에서 음악의 맥락과 상황 및 질감을 반영한 생성 예술적 비주얼을 디자인할 수 있었다.

공연 시스템의 측면에서는 사용 소프트웨어 환경을 Max/MSP 기반의 프로그램으로 통일하여 실제 공연 준비 및 실연 과정에서 비주얼라이징 시스템을 운용하며 발생할 수 있는 시스템적 위험요소를 최소화 할 수 있었다.

동국대학교 DONGGUK UNIVERSITY 1906

IV. 결론

본 논문은 노이즈 알고리즘 및 생성 예술 기반의 멀티미디어 작품 제작에 관한 연구이다. 노이즈가 다양한 매체에서 사용되는 방식을 탐구하고 이를 이용해 음악과 영상을 제작하였으며 흑-백, 사인파-노이즈의 도식이 주된 기법으로 사용되는 미니멀리즘 양식에서 벗어나 글리치, IDM 스타일에 어울리는 새로운 영상 제작 기법을 고찰하게 되었다. 더불어 음악과 영상의 인터랙션을 효율적으로 수행하기 위한 다양한 분석 기법들을 적용한 생성 예술 기반의 비주얼라이징 시스템을 디자인하여 영상이 음악을 보조하는 형태의 한 방향 인터랙션의 형태를 벗어나 음악과 영상이 합쳐져 작품 전체의 주제를 표현하는 다감각적인 멀티미디어 작품을 제작하고자 하였다.

영상과 음악이 일체화된 작품을 제작하기 위해서는 동시에 모든 요소의 변화량을 세심하게 확인하고 수정하는 과정이 필요하다. 본 연구에서는 스테레오 오디오에서 복잡한 리듬 정보를 추출하는 소프트웨어와 음원 전체 주파수 스펙트럼을 분석하여 악상의 변화를 데이터로 추출하는 소프트웨어를 제작하였으며 이러한 Max for Live 소프트웨어들의 도움으로 음악과 영상을 동시에 제작하는 것이 가능했다.

제작된 시스템을 이용하여 실제 작품으로 완성하는 과정에서는 다양한 시스템적 오류를 해결하고 최적화하는 과정이 필요했다. 실시간 렌더링을 이용해 제작되는 작품, 특히 생성 예술적 알고리즘에 기반한 작품은 실제 알고리즘의 적용 효과를 극대화하는 것도 중요하지만 작가의 의도를 온전히 담는

방법을 연구하는 것이 필수적이다. 전체적인 작품의 흐름을 조망하고 각 장면에 맞는 영상을 디자인하고 제어하면서 프로그램상에서 다양한 문제를 해결해야 하는 상황이 자주 발생하기 때문에 개발 시스템을 한 가지 환경으로 일원화하는 것이 중요하다. 본 논문의 작품 및 연구에서는 오디오, 비디오 및 3D 그래픽스 등 멀티미디어 작품에 필요한 요소들을 모두 Max 기반의 환경인 Max for Live와 Jitter를 이용해 구현함으로써 문제 해결 과정의 복잡도를 최소화 할 수 있었으며 해당 연구 산출물을 상용 소프트웨어로 개발하는 등 앞으로의 발전 가능성 또한 확인할 수 있었다.

하지만 이번 연구를 통해 발견한 앞으로의 개선점은 다음과 같다. 첫 번째는 연주자 및 오퍼레이터가 음악을 소프트웨어로 연주하는 것뿐만 아니라 영상 제어에 관련된 다양한 파라미터를 동시에 제어해야 했기 때문에 관객들의 시선에서는 연주자의 움직임과 음악 간의 상관관계를 찾아보기 힘들었다. 이를 개선하기 위해서는 연주자의 컨트롤러 조작이 실제 작품과 어떠한 관계가 있는지 좀 더 명확하게 보여줄 수 있는 연주용 인터페이스를 새로 개발하거나 또는 무대 환경을 새로 디자인하여 공연 시스템에 추가하는 것이 필요하다. 두 번째로는 원격 제어에 관련된 요소이다. 리듬 데이터를 전송하는데 사용한 OSC 데이터의 경우 용량이 그렇게 크지 않았기 때문에 전송 과정에서 지연(latency)이 매우 적게 발생했다. 그러나 오디오 서버에서 음원 전체의 주파수 스펙트럼을 분석한 데이터는 전송을 위해 데이터의 해상도를 매우 낮게(256x256픽셀의 이미지) 줄인 채로 전송을 해도 100~300ms 이내의 지연이 발생하여 개발 단계에서 구상했던 주파수 스펙트로그램

(spectrogram)을 사용한 실시간 오디오 스펙트럼 시각화 및 효과를 제외하게 되었다.

세 번째로는 오퍼레이터의 모니터링과 관련된 문제이다. 그래픽 서버에서 출력되고 프로젝터에 맺히는 메인 스크린의 영상을 오퍼레이터의 시점에서는 바라볼 수 없었다. 이를 해결하기 위해 공연 당일 그래픽 서버에서 다시 무선으로 오퍼레이터의 오디오 클라이언트가 설치된 노트북으로 비디오 데이터를 보냈지만, 위에서 언급한 네트워크 문제로 공연자의 시점에서 실시간으로 최종 출력물을 모니터링할 수 없었다. 다음 연구에서는 새로운 전송 프로토콜 또는 비디오 신호 자체를 유선으로 보내는 방식 등을 사용하여 네트워크 지연 현상을 개선한 시스템 디자인, 무대 시스템 개발 등의 연구가 필요할 것으로 보인다.

Keyword(검색어)
컴퓨터음악(computer music), 생성 예술(generative art), 오디오 시각화(audio visualization), 멀티미디어 음악(multimedia music), Max for Live, Max, MSP, Jitter, OpenGL

E-mail: unohee.official@gmail.com

참 고 문 헌

1. 단행본

- Alessandro Cipriani, & Maurizio Giri. (n.d.). 「Electronic Music and Sound Design」, (Contemponet, 2013)

- Andy Farnell. 「Designing Sound」, (The MIT Press, 2010)

- Charles Dodge, Thomas A. Jerse, 「Computer Music: synthesis, composition and performance, Second Edition」, (Schirmer Books, 1997)

- Curtis Roads, 「The Computer for music」, (MIT Press, 1996)

- Daniel Shiffman, 「The Nature of Code」, (Daniel Shiffman, 2012)

- Rick Snoman. 「Dance music manual」, (Focal Press, 2014)

- V.J Manzo, Will Kuhn. 「Interactive Composition: Strategies using Ableton Live and Max for Live」, (Oxford University Press, 2015)

2. 참고논문

- 라지웅, 「Max/MSP 와 Generative Art 를 이용한 멀티미디어음악 제작 연구」 (동국대학교 영상대학원 멀티미디어학과, 2018)

- 박상현, 「제너레티브 아트의 미학적 맥락과 기술적 유사 영역에 대한 연구」 (디지털디자인학연구), 12(2), 179-189

- 조환희, 「베이스 트롬본과 피아노의 실시간 사운드 프로세싱을 이용한 멀티미디어작품 제작 연구」(동국대학교 영상대학원 멀티미디어학과, 2019

- 피정훈,「실시간 음향 분석을 통한 3D Visualization 연구」(동국대학교 영상대학원 멀티미디어학과, 2008)

- Charles, J. n.d. A Tutorial on Spectral Sound Processing Using Max/MSP and Jitter. Computer Music Journal, 32, pp. 87-102.

3. 웹사이트

- Ableton
  https://www.ableton.com

- CCRMA: Karplus-strong Algorithm
  https://ccrma.stanford.edu/~jos/pasp/Karplus_Strong_Algorithm.html.

- Cycling74
  https://cycling74.com

- 백남준 아트센터: 참여 TV
  https://njp.ggcf.kr/참여-tv/

ABSTRACT

A Study on Audiovisual interaction system Design based on noise algorithms

Oh, Heewon

Department of Multimedia

Graduate School of Digital Image and Contents

Dongguk University

Audiovisual art or visual music refers to a work that attempts to express music in a visual form. With the development of technology in the late 20th century, visualization techniques using 3D graphics began to be used in works to capture not only auditory or visual experiences but also multi-sensory experiences in modern art and electronic music.

These audiovisual works are fundamentally created based on data visualization techniques and sonication techniques.

This study focused on two keywords, generative art and noise. Generative art refers to a work of art in which an autonomous system, not a human, directly determines the characteristics and shape of the work, and the result or process produced by the worker through a specific algorithm or system. In other word, generative art means "a work of art is created by a system through planned or programmed coordination."

In general, noise is a word that means noise or arbitrary visual stimulation, a form that does not show consistent colors or shapes in images or an aperiodic sound in acoustic term. Also, noise has been used as a term of the techniques that arbitrarily implements different random patterns depending on the medium.

In computer graphics, noise was used to implement the randomness, or fractal form, in which it was seen in the natural world. Using noise algorithms such as Perlin noise and simplex noise, it is used as the basis for generating abstract artworks represented by generative art. And also, it is used for creating topographic textures in real time, or making the texture of virtual objects more realistic. Likewise, auditory noise that is generally represented as white noise, has also been used

in techniques to transform artificial data to feel close to natural outputs, such as being used as a major component of physical model-based sound source synthesis incorporating audio filters and delays or to attenuate distortion caused by bit depth conversion of digital audio.

Among these various applications of noise, the purpose of the study is to develop a visualizing system that visualizes music in a three-dimensional space by combining audio spectrum data or transition detection technology based on a technique used as a main medium of computer-generated art.

부록: 첨부 DVD 설명

1. Event Horizon II 공연 영상 : 2021 년 11 월 13 일 이해랑 예술극장 공연 영상

2. Event Horizon II patch : 작품에 사용된 Max for Live, Jitter 패치 폴더