Journal of Digital Art Engineering & Multimedia Vol.7, No.3, September (2020), pp.187-198
ISSN 2508-9099(p) 2672-1198(e) *http://dx.doi.org/10.29056/jdaem.2020.09.01*

디지털아트 작품의 데이터 로깅 및 시각화를 위한 서버 개발

Server Development for Data Logging and Visualization of Digital Artwork

윤지현[1]

Ji-Hyun Yoon[1]

요 약

디지털 매체와 데이터 처리 기술의 발달은 예술 작품의 형식적, 내용적 변화를 이끈 요인으로 작용했다. 특히 디지털아트는 데이터의 형태로 상호 호환이 가능한 매체를 상호 연결하고, 그 체계와 방식을 바탕으로 예술적 실험을 실현한다. 본 논문에서는 디지털아트 작품이 실행되는 과정에서 생성 및 갱신되는 데이터를 주기적으로 데이터베이스에 저장하고, 이를 라인 차트와 레이더 차트로 시각화하는 서버를 설계하고 개발하였다. 서버는 디지털아트 작품과 네트워크 연결을 통해 독립적인 실행이 가능하도록 수평적 구조로 설계되었으며, 스케줄 설정에 의해 일정한 시간 간격으로 기록되는 데이터의 유형과 특정 상황에 따라 기록되는 이벤트 데이터 유형을 구별하여 데이터를 관리하고 차트로 출력하는 특징이 있다. 본 논문에서 제안한 서버는 관리와 보존의 측면에서 작품을 유지하고 이상 작동을 판단하는데 활용이 되고, 정보화 시대의 예술 작품의 측면에서 축적된 데이터를 통해 시대를 반영하고 기록하는 역할을 하게 된다.

핵심어 : 디지털아트, 데이터 로거, 데이터 시각화, 관리, 보존

Abstract

The development of digital media and data processing technology has acted as a factor that has led to changes in the form and content of art works. In particular, digital art connects interchangeable media in the form of data and materialize artistic experiments based on this system and method. In this paper, a server that periodically stores data generated and updated in the process of executing digital art works in a database and visualizes them in line charts and radar charts was designed and developed. This server was designed in a horizontal structure to enable independent execution through a network connection with digital art works. In addition, the server has a characteristic of managing and visualizing data by distinguishing the type of data recorded at regular time intervals and the type of event data recorded according to a specific situation. In terms of preservation and conservation, this server is used to maintain artworks and analyze abnormal behavior, and plays a role of reflecting and recording society and culture through accumulated data in the aspect of artworks in the information age.

Keyword : Digital art, Data logger, Data visualization, Preservation, Conservation

1 Division of Design and Art, Yonsei University, Wonju, Korea [Professor]
e-mail: ygdrasil@yonsei.ac.kr

Received(July 31, 2020), Review Result(1st: August 19, 2020), Accepted(September 4, 2020), Published(September 30, 2020)

1. 서론

1.1 연구의 배경 및 목적

디지털아트 작품은 고도화된 디지털 매체 기술을 보다 적극적으로 활용하면서 데이터를 기반으로 하는 '총체 예술 작품'(Gesamtkunstwerk)이다. 디지털아트 작품에서 문자, 이미지 그리고 소리 등은 미학적 의도나 목적에 따라 비트(bit)로 환원되어 서로 교환되거나 변환되어 새로운 복합된 감각의 방식으로 발현되기도 한다. 또한 인터넷을 비롯한 정보, 통신 기술의 발달로 데이터의 생산, 저장 그리고 전송이 용이해 지고, 대량의 데이터를 효율적으로 처리할 수 있는 컴퓨팅 기술의 발전으로 작품은 보다 복잡하고 융합된 양상으로 통합된다. 이러한 특징은 데이터를 기반으로 하는 디지털아트가 예술의 장르나 매체의 경계를 허물고 재매개 과정을 통해 창작자와 수용자의 관계를 새롭게 제시하는 계기를 마련하고 있다. 그러나 이러한 혼종화는 작품을 분석하고 보존 관리하는데 어려운 요인으로 작용하기 때문에 전통적인 예술 작품과는 다른 전략이 필요하다.

본 논문에서는 디지털아트 작품이 전시되어 실행되는 동안 생산 및 처리되는 데이터를 저장하고 시각화하는 시스템을 설계하고 구현한다. 제안하고자 하는 시스템의 연구 목적은 데이터 예술로서와 복원의 관점에서 디지털아트 작품의 본질인 디지털 데이터를 기록하여 데이터의 관점에서 분석하고 관리하는데 있다. 디지털아트 작품의 주된 특징으로 거론되는 상호작용성에서, 특히 작품을 구성하는 요소와 요소 사이, 작가가 창작한 작품과 관객 사이에서 입출력되는 데이터를 축적하고 이를 관리와 보존의 수단으로 활용하는 방안을 모색하기 위함이다. 전통적인 미술 작품을 개괄적인 정보로 기록하고 관리하는 방식에서 한발 더 나아가 작품 자체가 실시간으로 생성해 내는 의미 있는 데이터를 자료화 하고 보존하는 것은 작품 전반의 프로세스를 기록하고 이해하는 행위와 같다. 이를 통해 작품이 전시되는 시점이나 장소의 차이에서 비롯된 전반적인 결과물의 차이, 관람객의 참여에 따른 즉각적인 결과물의 차이 등을 파악할 수 있는 근거가 될 것이다. 작품의 데이터를 구조화하여 작품을 구성하는 매체와 시스템의 형식을 분석하는 것에 그치지 않고 시스템을 관통하는 작가의 의도와 동시대의 사회, 문화적 현상을 기록하고 분석하는 것과도 맞닿아 있기 때문이다. 센서나 인터넷에서 구한 실시간으로 변화되는 데이터가 작품의 결과에 반영이 된 후 사라지는 것과 이것을 순차적으로 기록한 누적 데이터의 가치는 분명히 다르다. 로이 에스콧(Roy Ascott)이 주장하는 바와 같이, 나비가 이 꽃에서 저 꽃으로 날아다니며 꽃가루를 퍼뜨리고 꿀을 얻듯 인간도 현실세계와 인터넷에서 다양한 네트워크에 접속하여 정보와 의미를 퍼뜨리고 축적해야 한다[1]는 것이다. 즉 꽃가루를 퍼뜨리는 역할 만큼 어떤 의미를 담은 꽃가루가 퍼뜨려 졌으며 그것이 어떤 형태로 진화하는지를 추적하고 관찰하는 행위 또한 중요하다.

1.2 연구의 방법

항공기의 블랙박스나 기상 또는 환경 데이터를 기록하기 위한 데이터 로거(Data logger)는 획득하여 분석하고자 하는 정보의 대상이 특정지어 지지만, 디지털아트 작품의 데이터를 기록하기 위한 시스템은 관리의 주체가 각 개별 작품에 따라 데이터를 어떻게 구조화 할지를 판단해야 한다. 따라서 다양한 작품에서 공통적으로 적용될 수 있는 기능을 정의하고 최소한의 데이터베이스 자료구조(Database schema)를 제안한다. 이를 바탕으로 개발된 서버 시스템은 안정적인 구동을 검증하고 가상의 작품과의 연동을 테스트하여 그 결과를 기술한다. 본 연구를 통해 구현된 서버와 테스트용 코드 파일은 깃허브(https://github.com/ygdrasilcave/DL4MAP)에서 확인할 수 있으며, 작품에 적용하는 목적에 따라 변경 및 응용이 가능하다.

2. 데이터 로깅 및 시각화를 위한 서버

2.1 데이터 로깅 서버의 기능과 설계

디지털아트 작품의 데이터 로깅 및 시각화를 위한 서버는 다음과 같은 기능을 지원하도록 설계하였다.

2.1.1 서버의 독립 실행

본 논문은 다양한 디지털 매체의 활용으로 통합적이고 상호연결의 특징을 띠는 디지털아트 작품을 위해 네트워크 연결을 통한 독립된 데이터 로깅 시스템을 제안한다. 이 시스템은 작품, 서버 그리고 시각화 서비스를 포괄적으로 포함하며, 작품과 별도로 개별 실행이 가능한 서버는 작품 자체의 예술적 목표와 의도를 유지하도록 설계한다. 따라서 작품과 서버는 OSC [2] 또는 MQTT [3] 프로토콜을 이용한 통신으로 데이터를 송수신하게 되며 상호 종속적인 체계를 지양한다.

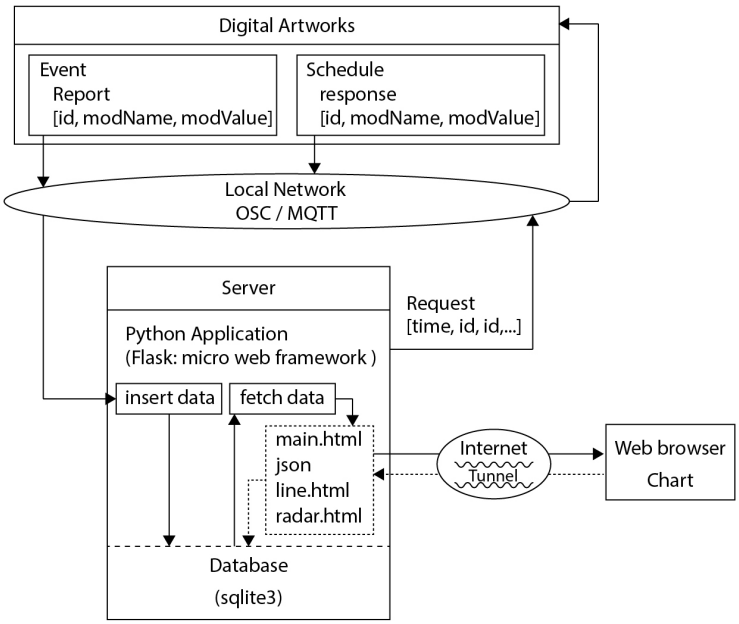
2.1.2 스케줄 관리에 의한 데이터 요청과 기록

작품에서 생성된 데이터를 기록하는 시점은 사용자가 지정한 일정과 이벤트에 의해 요청(Request) 또는 보고(Report)되는 방식으로 나누어진다. 디지털아트 작품의 전시기간 동안 일정하게 변화되거나 유지되어야 하는 데이터를 추적 관찰하고, 센서에 감지된 수치나 관람객이 입력한 정보를 기록하도록 한다. 스케줄 데이터는 서버 상에서 일정과 시간이 관리되어 작품에 데이터를 요청하여 반환된 값을 데이터베이스에 기록한다. 그러나 이벤트 데이터는 특정 상황에 따라 작품에서 서버로 전송되는 데이터로 그 유형을 구별하도록 한다.

2.1.3 데이터 분석을 위한 시각화 및 JSON 파일 지원

데이터베이스에 기록된 모든 데이터는 수치형 데이터를 위한 라인 차트(Line Chart)와 범주형 데이터를 위한 레이더 차트(Radar Chart)로 시각화 되며, 작품을 관리하고 분석하는 것에 그 목적이 있다. 또한 데이터의 활용성을 확장하기 위해 JSON 포맷의 파일로 추출이 가능하다. 이 기능은 작품이 전시되는 시점에 입력된 데이터를 다시 작품에 입력하여 그 결과를 되돌려 볼 수 있도록 하는데 목적이 있으며, 서로 다른 장소나 기간 동안 수집된 데이터와 상호 비교하여 분석하는데 활용이 가능하다.

앞에서 밝힌 기능들은 제안하고자 하는 서버의 단독 실행만으로 온전한 작용을 하지 못한다. 디지털아트 작품의 제작에 있어서 서버와 작품 간의 네트워크 통신을 위한 부가적인 작업과 조치는 반드시 필요하다. 그러나 데이터 예술로서의 본질에 대한 인식과 디지털아트 작품이 직면해 있는 관리와 보존에 대한 문제에 더욱 적극적으로 대처하며 준비하기 위한 방안의 모색은 필요하다. 따라서 작품이 실행되는 동안 생성되고 변형 또는 갱신되는 유의미한 데이터를 축적하고, 데이터 분석을 통한 보다 적극적인 관리 방법론의 기틀을 마련하고자 설계한 서버와 전체 시스템의 구성은 [그림 1]과 같다.



[그림 1] 디지털아트 작품의 데이터 로깅 및 시각화를 위한 서버

[Fig. 1] A server for data logging and visualization of digital artwork

2.2 서버의 구현과 데이터 시각화

디지털아트 작품과 연동되는 서버는 라즈베리파이(Raspberry pi 3 Model B+)를 이용하여 구현하였다. 공식 운영체제인 라즈비안(Raspbian Buster) 리눅스와 SQLite3 [4]를 설치하고, 테스트에 필요한 데이터베이스를 생성하였다. 데이터베이스는 작품에 따라 관리자가 직접 생성 및 설정하여 사용하게 되고, 테이블은 작품이 실행되는 기간 동안 매일 자동으로 생성 된다. 각 테이블은 두 자리의 연도, 월, 일(YYMMDD)로 구분되며, 저장되는 자료의 구조는 [그림 2]와 같이 구성하였다. "logTime"은 데이터가 저장되는 시각을 두 자리의 시, 분, 초(hh:mm:ss)로 기록하게 되고, "schedOrEvt"는 스케줄(sched) 또는 이벤트(evt)에 의한 기록 여부를 구분하기 위한 항목이다. "modID", "modName", "modValue"는 작품을 구성하는 요소인 소프트웨어, 전자부품 등을 ID와 이름으로 구별하여 전송된 데이터의 값을 기록하기 위한 항목이다. "modValue" 항목은 데이터의 형태가 텍스트로 지정 되어 있는데, 수치형 데이터일 경우 부동소수점을 가지는 숫자를, 범주형 데이터의 경우 정수를 텍스트의 형태로 입력이 가능하다. 수치형 또는 범주형 데이터는 ID의 범위에 따라 구분되는데, ID 1번에서 99번 까지는 수치형 데이터, 100번 이상은 범주형 데이터로 취급하게 된다.

```
create table TB_project_200706 (
    logTime text,
    schedOrEvt text,
    modID integer,
    modName text,
    modValue text
);
```

[그림 2] 데이터베이스 자료의 구조

[Fig. 2] database schema

서버는 파이썬(Python) 언어 기반의 플라스크(Flask) 웹 프레임워크 [5]를 활용하여 개발하였으며, HTTP 프로토콜 GET 방식의 요청에 응답하기 위한 웹서버의 기능을 8181 포트로 지정하여 구성하였다. 또한 스케줄 관리 및 SQLite3 데이터베이스와 디지털아트 작품과의 연동으로 데이터를 저장하고 저장된 데이터를 JSON 파일로 반환하는 역할을 수행한다. [그림 3]은 관리자가 데이터 시각화 웹페이지 URL을 요청할 때 실행되는 각 함수에 대한 정의로, "/"는 "main.html", "/line"은 "line.html", "/radar"는 "radar.html" 페이지를 호출하고, "/json"은 json.json 파일을 출력하여 저장할 수 있도록 한다.

```
@app.route("/line", methods=['GET'])
def line():
    return render_template('line.html')
@app.route("/radar", methods=['GET'])
def radar():
    return render_template('radar.html')
@app.route("/json", methods=['GET'])
def table():
    return jsonify(fetchTable())
@app.route("/")
def main():
    tableName = TABLE_NAME_PREFIX + getTableName()
    serverIPport = request.host
    serverIP = serverIPport.split(':')[0]
    tableList = getTableList()
    ips = []
    ips.append('http://' + serverIPport + '/?tableName=' + getTableName())
    ips.append('http://' + serverIPport + '/json?tableName=' + getTableName())
    ips.append('http://' + serverIPport + '/line?serverIP=' + serverIPport + '&dbTableName=' + getTableName())
    ips.append(request.remote_addr)
    return render_template('main.html', readings=fetchTable(), tableName=tableName, ips=ips, info=info, tableList=tableList)
```

[그림 3] 데이터 시각화를 위한 웹페이지 호출 함수

[Fig. 3] web page call function for data visualization

[그림 4]는 전시 일정에 따라 설정이 가능한 스케줄 함수이다. "schedCreateTable()" 함수는 전시가 시작되는 시간 15분 전에 실행되어 "createTable()" 함수를 호출하고 데이터베이스에 새로운 테이블을 생성하도록 한다. "schedUpdateDB()" 함수는 스케줄 관리 목록에 있는 모든 ID의 값을 리스트의 형태로 OSC 클라이언트(작품)에 데이터를 요청하고, 해당되는 ID의 클라이언트는 이에 대한 응답으로 데이터를 전송하여 DB에 기록되어 진다. 서버와 미디어아트 작품 사이에 송수신 되는 OSC 메시지의 구조는 테스트에 대해 기술한 다음 장에서 구체적으로 밝힌다.

```
@sched.scheduled_job('cron', day_of_week=sched_day_of_week, hour=sched_hour, minute=sched_minute)
def schedUpdateDB():
    now = datetime.now().strftime("%H:%M:%S")
    oscMsg_schedRequest_moduleID[0] = now
    for i in range(len(oscClient)):
        oscSendMsg(oscClient[i], oscMsg_schedRequest_moduleID)
        print('------ request ------')
        print(oscClientAddr[i])
        print(oscMsg_schedRequest_moduleID)
        print('--------------------')
@sched.scheduled_job('cron', day_of_week=sched_day_of_week, hour=_hour, minute=45)
def schedCreateTable():
    with app.app_context():
        now = datetime.now().strftime("%H:%M:%S")
        createTable()
        print('------ New Table created: ' + now)
```

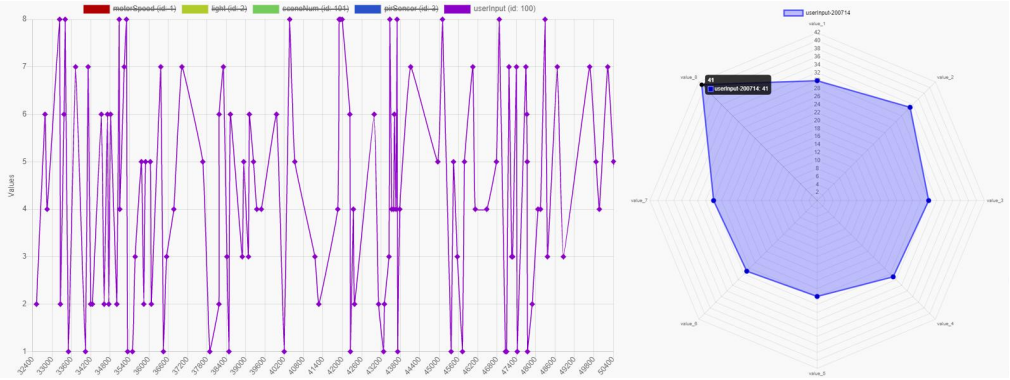
[그림 4] 스케줄에 따라 호출되는 함수

[Fig. 4] functions call on a schedule

이 논문에서 제안하는 서버는 작품에서 전송되는 모든 데이터를 부가적인 가공의 과정을 거치지 않고 그대로 저장하고, JSON 파일의 형태로 출력하기 때문에 이를 그래프로 표현하기 위한 준비가 필요하다. 라인 차트와 레이더 차트를 출력하는 웹페이지는 자바스크립트의 chart.js 라이브러리 [6]를 활용하였으며, 시각화를 위한 데이터 및 구조의 변경이 요구된다.

우선 데이터가 기록된 시간에 대한 데이터(logTime)는 초 단위로 변환하여 라인 차트 X축의 구간별 변화량을 일정하게 유지하고 데이터가 기록되는 시간 간격의 길고 짧음을 파악할 수 있도록

하였다. 이벤트 기반의 데이터는 일정한 간격으로 기록되는 것이 아니기 때문에 이벤트가 발생되는 시점 간의 간격을 그대로 유지하기 위함이다. 차트를 어떻게 디자인할지 결정할 때에 고려해야 할 또 한 가지의 요소는 기준선과 X축(가로)과 Y축(세로)의 척도 [7]인데, 라인 차트 Y축의 최솟값과 최댓값의 범위는 입력된 모든 데이터에 따라 달라지기 때문에 영점 기준선을 반드시 가지지 않는다. 전체 데이터의 각 값(modValue)은 점의 높이로 나타나며 스케줄 데이터(sched)인 경우 원형, 이벤트 데이터(evt)인 경우 사각형으로 서로 구별이 가능하다. 이 값들은 변환 없이 온전히 기록된 그대로 차트 상에 표시되는데, 꺾은선형 또는 계단형으로 정의해 줄 수 있다. 레이더 차트에서는 ID가 100번 이상인 범주형 데이터의 개별 값을 집산하고 동일한 값을 갖는 정수의 입력 횟수를 반환하여 차트로 출력한다. 범주형 데이터를 정수의 텍스트 형태로 입력하도록 한 이유가 여기에 있으며 명목 변수로 다루기 위해서이다. [그림 5]는 동일한 데이터를 두 종류의 차트로 출력한 것인데, 왼쪽은 일정한 시간의 간격을 기준으로 데이터가 기록된 빈도를 파악할 수 있으며, 오른쪽은 같은 값을 가지는 명목 변수가 입력된 횟수를 상대적으로 파악하는 데 사용할 수 있다.

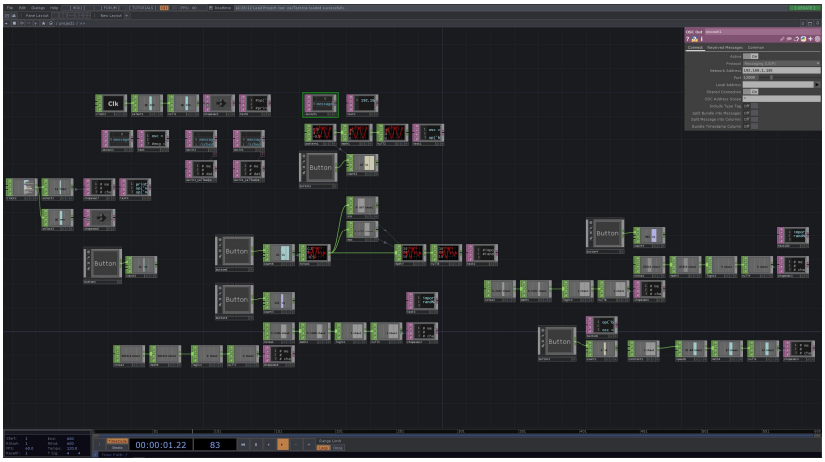


[그림 5] 동일한 데이터에 대한 라인 차트와 레이더 차트

[Fig. 5] line chart and radar chart for the identical data

3. 서버의 테스트 및 평가

구현된 서버는 1개월의 테스트 기간 동안 24시간 구동되었으며, 디지털아트 작품을 대신할 응용 프로그램은 터치 디자이너(Touchdesigner)를 이용하여 [그림 6]과 같이 프로그래밍한 후 테스트를 진행하였다. 기록된 데이터를 확인하기 위해서는 서버가 연결되어 있는 동일한 로컬 네트워크에 연결된 웹브라우저를 통해 가능하지만, ngrok [8] 등의 터널링(Tunneling) 서비스를 활용하면 미술관이나 전시장의 네트워크 방화벽 외부에서도 접근이 가능해 진다. 테스트를 위해 적용한 설정과 OSC 프로토콜을 요약하면 [표 1]과 같다.



[그림 6] 테스트를 위한 응용 프로그램

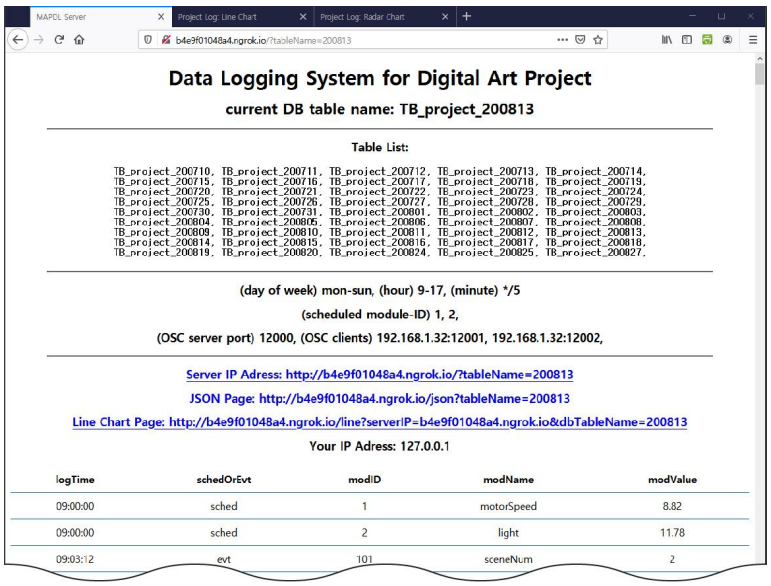
[Fig. 6] Application for testing

[표 1] 테스트를 위한 설정

[Table 1] Setup for testing

구분		설정	설명
작품에 부여된 ID (modID)		1, 2, 3, 100, 101	5개의 ID를 부여함
작품 구성 요소의 명칭 (modName)		'motorSpeed', 'light', 'pirSensor', 'userInput', 'sceneNum'	motorSpeed: 모터의 속도 (ID 1번) light: 조명의 밝기 (ID 2번) pirSensor: 인체 감지 센서 (ID 3번) userInput: 관람객 입력 값 (ID 100번) sceneNum: 장면 번호 (ID 101번)
코드 상수	DATABASE	'testDB'	테스트에 사용된 데이터베이스 명
	TABLE_NAME_PREFIX	'TB_project_'	테이블 명 앞에 붙는 단어로, 서버에서 갱신되어 연결되는 날짜와 함께 테이블 명이 됨
	sched_day_of_week	'mon-sun'	월요일부터 일요일 까지 스케줄 실행
	sched_hour	'9-17'	9시부터 18시 이전 까지 스케줄 실행
	sched_minute	'*/5'	5분 단위로 스케줄 실행
	oscServer_port	12000	서버에서 동작하는 OSC 서버 포트
	oscClientAddr	[('192.168.1.32', 12001), ('192.168.1.32', 12002)]	OSC 클라이언트의 IP와 포트로 2개 지정
OSC 프로토콜	OSC address pattern	'/schedRequest'	스케줄에 의한 데이터 요청
		'/schedReport'	스케줄 요청에 대한 데이터 전송
		'/evtReport'	이벤트 데이터 전송
	OSC type tag	'iss'	작품에서 전송되어 서버에서 수신되는 modID(integer), modName(string), modValue(string)
		'sii'	서버에서 스케줄 데이터 요청시 전송되는 '요청시간(string)과 요청할 ID 2개(integer, 1번과 2번)

미디어아트 작품의 관리자가 웹브라우저의 URL 필드에 서버의 IP주소와 포트를 입력하면 "main.html" 페이지가 출력되어 [그림 7]과 같이 나타난다. 메인 페이지에서는 서버의 현재 날짜를 확인한 후 해당되는 최신 테이블의 데이터를 페치(fetch)하여 표의 형태로 출력한다. 이 페이지에서는 데이터베이스에 저장된 데이터 이외에 기본적인 설정 정보에 대한 내용도 확인이 가능한데, 생성된 모든 테이블 리스트, 스케줄 설정, 스케줄 설정된 모듈의 ID, OSC 클라이언트의 설정 정보 등이 있다. 그리고 JSON 파일을 저장할 수 있는 경로와 라인 차트 페이지의 링크 경로도 출력하기 때문에 마우스 클릭으로 이동할 수 있다.

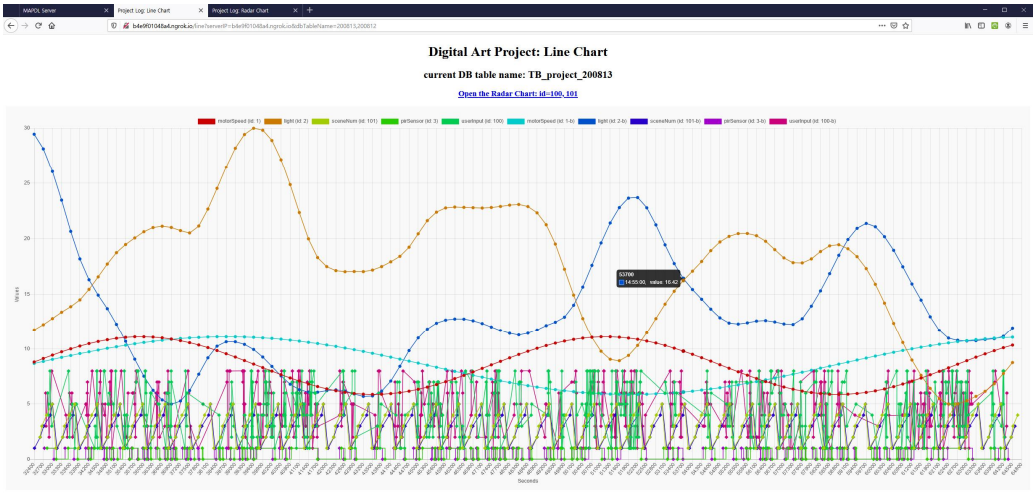


[그림 7] 메인 페이지

[Fig. 7] Main page

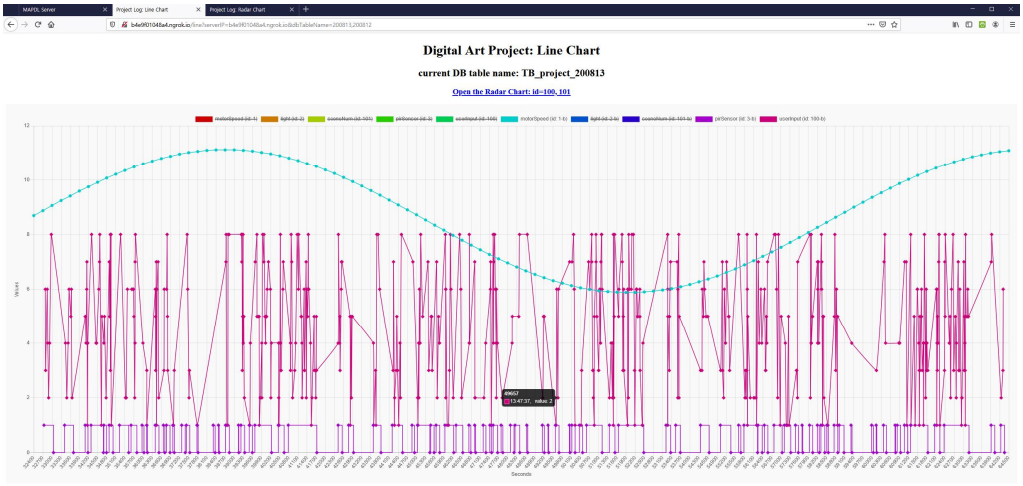
각 웹페이지는 공통적으로 "tableName=200813"과 같이 쿼리 문자열(Query string)로 테이블을 지정하거나 새로운 테이블로 변경해 데이터를 다시 불러오는 것이 가능하다. 각 테이블은 매일 설정된 전시 시작 시간의 15분 전, 날짜별로 생성됨으로 다른 날짜의 데이터를 확인하고자 하는 경우 이 값을 원하는 테이블의 이름으로 할당하여 URL 필드에 추가할 수 있다. 또한 서로 다른 날짜의 데이터를 비교해 보고자 할 경우, 최대 2개의 테이블 명을 "tableName=200813,200812"와 같이 입력하면 [그림 8]과 같이 대조가 가능하다. 이러한 데이터의 시각화는 각 데이터의 변화와 그 정도를 파악하여 실제 작품에서 표현되는 요소들의 상태를 파악하거나 상관관계를 드러내는 역할을 한다. [그림 9]의 예와 같이 'pirSensor'의 값이 1인 상태에서만 'userInput'의 값이 변하는 것을 확인할 수 있는데, 이는 인체감지 센서가 1인 상태는 관람자가 작품을 감상한 시간을 담고 있으며, 그 시간

동안 관람자가 선택한 1에서 8사이의 어떠한 값(명목 변수)을 기록한 테스트 모델이다. 그러나 궁극적으로 이러한 시각화를 위해서는 어떤 데이터를 선택하여 어떤 주기로 기록하는가에 따라 그 기능성과 의미하는 바가 달라질 수 있기 때문에 작품 관리에 필요한 데이터의 집합 구조를 목적에 맞도록 설계하는 것이 중요하다. 조직화 할 수 없는 데이터는 정보 디자인으로서의 가치를 가질 수 없으며 [9], 데이터베이스 자료의 구조에 따라 기록된 데이터의 의미와 가치는 달라질 수 있다.



[그림 8] 라인 차트 페이지

[Fig. 8] Line chart page

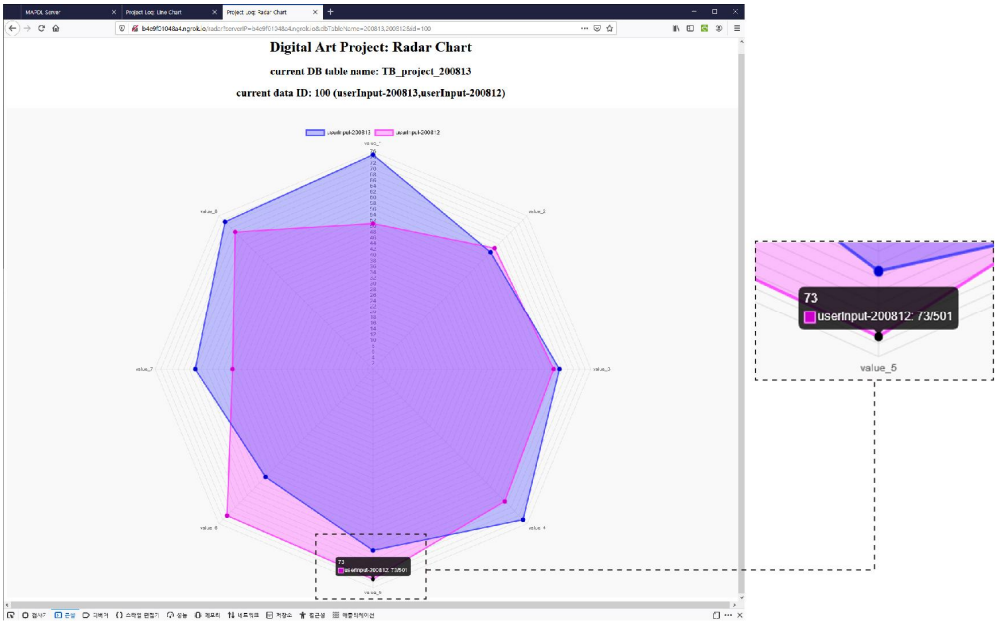


[그림 9] 라인 차트 페이지에서 선택한 데이터 표시

[Fig. 9] Show selected data on the line chart page

[그림 10]은 서로 다른 날짜의 데이터를 유형으로 분류하여 각 유형별 입력 또는 선택된 횟수를

나타낸 레이더 차트 모델이다. 이 유형 값은 정량적 가중치가 없고 각 유형을 구분하기 위해 사용되었으며, 오히려 누적 입력 횟수가 중요하다. 라인 차트와 레이더 차트 모두 마우스 커서를 데이터의 실제 값을 나타내는 점에 위치시키면 상세정보 창이 나타나는데, 레이더 차트에서는 전체 데이터의 입력 횟수와 현재 유형의 입력 횟수를 수치로 확인 가능하다. 라인 차트에서는 초 단위로 변환된 X축의 척도가 시:분:초(hh:mm:ss)로 나타나 실제 기록된 시간을 파악하는데 용이하다.



[그림 10] 레이더 차트와 상세정보 창

[Fig. 10] Radar chart and pop-up(tooltip) window

테스트 기간 동안 서버는 안정적으로 동작하였으며 일 평균 1100-1200개의 행(row)이 기록 되었고, 생성된 데이터베이스의 파일 크기는 약 960KiB로 나타났다. 이러한 결과는 스케줄 데이터의 저장 간격의 설정과 이벤트 데이터의 저장 빈도에 따라 차이가 생긴다. 축적된 데이터의 시각화는 관리자가 작품의 동작 패턴을 파악하거나 이상 현상에 대처할 수 있도록 하며, 작품의 실행 과정을 복기하는 목적으로 사용이 가능하다. 그러나 데이터의 가독성을 높이기 위해 시간이나 변수의 범위를 조정할 수 있는 기능의 추가가 필요해 보인다.

4. 결론

본 논문에서는 디지털아트 작품에 의해 생성, 갱신되는 데이터를 기록하고 시각화하는 서버 시스템을 제안하였다. 총체 데이터 예술로서 데이터를 소비하는 것에 머무는 것이 아니라, 데이터의

변화 과정을 추적하고, 수용자와의 상호작용에 따라 입력된 정보를 기록하여 이를 작품의 관리와 분석에 활용하는 예시를 보여주고자 하였다. 디지털아트 작품과 서버는 수평적 구조로 연결되어 데이터를 송수신하고, 스케줄과 이벤트에 의한 데이터베이스 저장 기능이 갖추어져 있다. 그리고 시간의 흐름에 따라 누적된 데이터는 라인 차트 또는 레이더 차트로 시각화 되어 작품의 실행과 관련된 데이터의 흐름과 상황에 대한 직관적인 판단이 가능하도록 하며, 이러한 시지각 중심의 패턴인식은 효율적이며 또한 즉각적이다. [10] 서버에서 출력되는 JSON 파일은 데이터 분석, 패턴 분석 등에 활용이 가능하고 작품을 복기하는 데이터로 사용할 수 있다. 디지털아트 작품은 알고리즘이나 데이터를 다루는 복합된 시스템, 관람자의 개입과 상호작용에 의해 완성되기도 한다. 이러한 이유 때문에 디지털아트 작품에서 데이터의 저장과 보존은 중요한 의미를 지닌다. 제안된 서버는 파이썬 언어를 기반으로 개발되었기 때문에 운영체제에 관계없이 설치와 실행이 가능하다는 장점이 있다. 그러나 차트로 출력되는 데이터의 범위를 선택하거나 특정 데이터를 검색할 수 있는 기능이 요구되어 이를 보완하는 연구를 진행할 예정이다.

References

[1] T. H. Syn, “Roy Ascott’s Techno-ethic Arts and World Art Tendencies”, Journal of Korean Society of Dance Science, vol. 34, no. 4, December 2017, pp. 83-93, doi: 10.21539/Ksds.2017.34.4.83.

[2] “Introduction to OSC”, opensoundcontrol.org, http://opensoundcontrol.org/introduction-osc, (accessed August 10, 2020).

[3] “MQTT: The Standard for IoT Messaging”, mqtt.org, https://mqtt.org/software/, (accessed August 10, 2020).

[4] “What Is SQLite?”, sqlite.org, https://www.sqlite.org/index.html, (accessed August 10, 2020).

[5] “Flask Documentation”, palletsprojects.com/p/flask/, https://flask.palletsprojects.com/en/1.1.x/, (accessed July 2, 2020).

[6] “Chart.js”, chartjs.org, https://www.chartjs.org/, (accessed July 2, 2020).

[7] A. Cairo, The truthful art (Korean Edition), Insight, 2019.

[8] “ngrok”, ngrok.com, https://ngrok.com/, (accessed August 16, 2020).

[9] J. K. Kim, “A Study for Foundation Properties of Information Visualization in Information Design”, Journal of Digital Design, vol. 9, no. 3, July 2009, pp. 313-324, doi: 10.17280/jdd.2009.9.3.030.

[10] J. W. Park and H. Y. Kim, “Review of Artistic Data Visualization”, Journal of Digital Design, vol. 11, no. 3, July 2011, pp. 193-202, doi: 10.17280/jdd.2011.11.3.019.