DOI : 10.5392/JKCA.2011.11.2.113

Eclipse기반 GMF 기법을 이용한 미디어 아트 콘텐츠 저작도구 개발

Development of the Media Art Contents Authoring Tool Using Eclipse-based GMF Technique

곽재호, 박송림, 김회율
한양대학교 전자컴퓨터통신공학과

Jae-Ho Kwak(jhkwak@vision.hanyang.ac.kr), Songlin Piao(slpiao@vision.hanyang.ac.kr), Whoi-Yul Kim(wykim@hanyang.ac.kr)

요약

본 논문에서는 미디어 아트 콘텐츠 저작도구인 ECAS(Exhibition Contents Authoring System)와 그 개발 방법을 소개한다. ECAS는 기존의 저작도구가 가진 텍스트 기반의 프로그래밍 방식의 문제점을 개선하기 위해 그래픽 사용자 인터페이스를 기반으로 개발되었으며, 그래픽 사용자 인터페이스의 효율적인 구현을 위해 Eclipse가 제공하는 GMF(Graphical Modeling Framework) 기법을 사용하였다. ECAS의 속도 및 성능 향상을 위해 SWT(Standard Widget Toolkit), JIT(Just-In-Time)와 같은 속도 향상 기법들을 활용하였다. 얼굴 검출 기능, 사운드 효과 기능, 하드웨어 연동 기능 등의 주요 기능들을 중심으로 기존의 저작도구와 작품 구현 비교를 통해 ECAS의 편리성과 효율성을 보였다.

■ 중심어 : | 미디어 아트 콘텐츠 저작도구 | 이클립스 | 그래픽모델링프레임워크 |

Abstract

We introduce a media art contents authoring tool called ECAS(Exhibition Contents Authoring System), and methodology employed in its development. ECAS was developed using the graphical user interface to overcome difficulties with existing authoring tools that are text-based. For efficient implementation of graphical user interface, GMF(Graphical Modeling Framework) technique by Eclipse was used. SWT(Standard Widget Toolkit) and JIT(Just-In-Time) were used to improve speed and performance of ECAS. We show convenience and efficiency of ECAS by comparing with existing media art contents authoring tools.

■ keyword : | Media Art Contents Authoring Tool | Eclipse | Graphical Modeling Framework |

I. 서 론

1. 연구의 배경

현대 예술의 한 주류인 디지털 미디어 아트는 디지털 테크놀로지와의 기술 융합을 통해 빠르게 발전하고 있다. 특히 미디어 아트와 디지털 테크놀로지의 융합을 통해 컴퓨터 및 디지털 관련 기술은 인터랙티브 미디어 아트를 중심으로 예술의 새로운 관심사로 등장하고 있

* 본 연구는 문화체육관광부 및 한국 콘텐츠진흥원의 2009(2010)년도 문화콘텐츠산업기술지원사업의 연구결과로 수행되었음
접수번호 : #100524-002
접수일자 : 2010년 05월 24일
심사완료일 : 2010년 11월 25일
교신저자 : 김회율, e-mail : wykim@hanyang.ac.kr

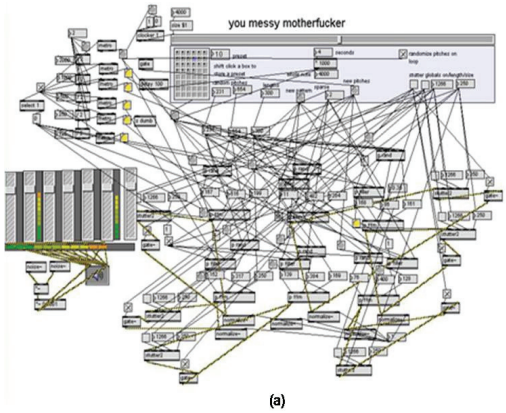
다[1]. 인터랙티브 미디어 아트는 다른 순수 예술과 달리 과학 기술과 예술이 접목된 다양한 형태의 콘텐츠를 선보일 수 있는 분야이며, 관객과 상호 소통할 수 있는 속성을 지니고 있기 때문에 전문적인 미디어 아트 전시 공간은 물론, 대중적 공간으로까지 그 영역을 넓히고 있다. 또한, 인터랙티브 미디어 아트는 전통적인 예술 매체와는 달리 새로운 환경과 기술요소를 바탕으로 구현되며, 환경과 기술이 다원화됨에 따라 관객과 아티스트 사이의 상호 소통 관계를 새롭게 형성하고 있다.

디지털 미디어 아티스들은 인터랙티브 미디어 아트 콘텐츠를 창작을 위해 미디어 아트 콘텐츠 저작도구(Media art contents authoring tool)라는 특수한 목적의 소프트웨어를 사용한다. 이 소프트웨어는 아티스트들이 콘텐츠를 창작할 때 예술적, 기술적 요소를 활용할 수 있도록 다양한 미디어 아트 콘텐츠 제작 기능을 지원한다.

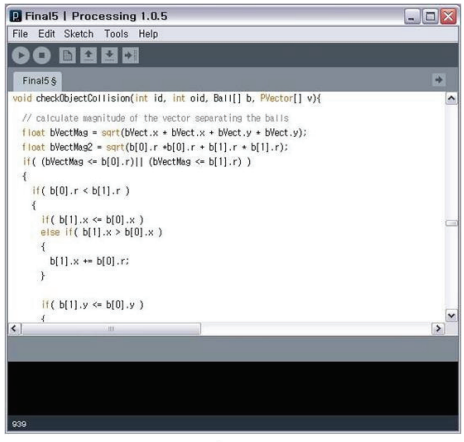
미디어 아트 콘텐츠 저작도구는 크게 두 가지 형태로 분류 된다. 그중 하나는, 그래픽 유저 인터페이스를 기반으로 한 형태로, 콘텐츠 제작에 필요한 기능들이 그래픽화 된 사각형의 기능 블록으로 제공된다. Max/MSP/Jitter[2], Openframeworks[3], VVVV[4] 등이 여기에 해당한다. 다른 하나는, 텍스트를 이용한 직접 프로그래밍 방식을 기반으로 하는 형태로, 주요 기능들은 내부 함수나 외부 라이브러리 형태로 지원된다. 이 경우 아티스트들은 프로그래밍에 대한 사전 지식을 가지고 있어야 하며, 미디어 아트 콘텐츠 제작을 위해 저작도구 상에서 직접 프로그래밍을 해야 한다. 대표적인 텍스트 기반의 저작도구로는 *Processing*이 있다[5].

기존의 저작도구들은 현재 미디어 아티스트들에 의해 많이 사용되기는 하나 몇 가지 문제점을 안고 있다. Max/MSP/Jitter는 그래픽 유저 인터페이스 방식을 채용하고 있으며, 많은 양의 기능을 제공하고 있어 다양한 콘텐츠 제작이 가능하다. 그러나 [그림 1(a)]에서처럼 기능이 매우 세분화 되어 있어, 저작도구에 익숙하지 않는 미디어 아티스트들이 사용하기에는 전문가 수준의 지식을 요구된다는 면에서 사용상의 어려움이 있다. 즉, 제공되는 기능 블록들이 명확하지 않아 도움말을 읽어보지 않고서는 직관적으로 기능을 파악하기가 어렵다. *Processing*의 경우, [그림 1(b)]처럼 JAVA 기반의 언어를 이용하여 직접 프로그래밍을 해야 하기 때문에 프로그래밍에 서툰 미디어 아티스트들이 사용하기 어렵고 활용이 제한적이라는 한계를 가지고 있다. 또한 아티스트가 필요로 하는 기능을 사용하려면 외부 라이브러리를 찾아서 추가해야하고, 이 또한 저작도구에 익숙하지 않은 아티스트들에게는 어려운 일이다.

따라서 콘텐츠 제작을 원하는 아티스트들의 입장에서 기술적 접근이 쉽고, 사용이 간편한 미디어 아트 콘텐츠 저작도구의 개발이 필요하다. 본 논문에서는 디지털 미디어 아트 콘텐츠 저작도구인 ECAS(Exhibition Contents Authoring System)와 그 개발 방법을 소개한다.



(a)



(b)

그림 1. 기존 저작도구의 문제점
(a) 기능이 과도하게 세분화 되어 있는 Max/MSP/Jitter
(b) 텍스트 기반의 프로그래밍이 필요한 Processing

2. 연구의 범위와 방법

ECAS는 기존 저작도구에서 아티스트들이 직접 프로그래밍을 해야 하는 어려움을 해결하기 위해 비주얼 그래픽 인터페이스 프로그래밍 환경을 기반으로 개발 되었다. 비주얼 그래픽 인터페이스의 효율적인 개발을 위해 Eclipse에서 제공하는 GMF(Graphical Modeling Framework) 기법[6][7]을 활용하였다. 또한, 기존 저작도구에서 너무 세분화 되어 있어 사용하기 불편했던 기능들을 콘텐츠 제작에 주로 사용되는 기능들을 중심으로 통합 구현하여, 사용하기 쉽고 편리하도록 하였다. 마지막으로 기존 저작도구들이 가지고 있지는 않지만 미디어 아티스트들이 필요로 하는 요소기술을 추가함으로써 미디어 아트 콘텐츠 저작도구로서의 기능에 더욱 충실하도록 하였다.

ECAS는 크게 세 가지 기능을 중심으로 개발되었다. 비디오 처리 기능, 사운드 처리 기능, 하드웨어 연동 처리 기능이 그것이다. 첫째, 비디오 처리 기능은 카메라와 같은 영상 입력장치로부터 입력된 영상에서 미디어 아트 콘텐츠 제작에 필요한 정보를 추출하고 이를 화면에 출력하는 기능이다. 예를 들어, 비디오 처리 기능은 아티스트가 관람객의 움직임에 따라 작품의 내용이 변경되는 콘텐츠를 만들고자 하는 경우 카메라로부터 입력된 영상으로부터 관람객의 움직임 정보를 추출하고 이를 콘텐츠 제작에 활용할 수 있도록 지원한다. 또한, 비디오 처리 기능은 관람객의 움직임뿐만 아니라 관람객 신체의 일부분(예를 들어 얼굴)을 검출하여 이를 활용한 콘텐츠 제작이 가능하도록 지원한다.

둘째, 사운드 처리 기능은 마이크와 같은 사운드 입력장치로부터 입력되는 오디오 신호를 이용하여 실시간으로 사운드를 생성하고, 생성된 사운드를 활용하여 사운드를 합성하거나 다양한 효과를 적용하여 이에 대한 결과를 출력하는 기능을 말한다. 미디어 아티스트들은 사운드 기능을 통해 사운드를 활용한 보다 폭넓은 미디어 아트 콘텐츠 창작 활동이 가능하다.

마지막으로 하드웨어 연동 처리 기능은 ECAS에서, 미디어 아트 콘텐츠 창작에 자주 이용되는 하드웨어를 보다 쉽고 편리하게 연동하여 사용할 수 있는 기능을 말한다. 미디어 아트 콘텐츠 제작에 필요한 터치 센서나 버튼과 같은 하드웨어를 단지 ECAS에 연결하는 것으로 해당 하드웨어를 연동하여 콘텐츠를 창작할 수 있다.

II. 관련연구

1. 미디어 아트 콘텐츠 저작을 위한 저작도구

기존에 미디어 아티스트들이 주로 사용하는 콘텐츠 저작도구로는 I.1절에 언급한 Max/MSP/Jitter, *Processing*, Openframeworks, VVVV 외에도 Adobe 사의 Flash[8], Pure Data와 Gem[9], 그리고 Quartz Composer[10]등이 있다. [그림 2]는 기존의 미디어 아티스트들이 주로 사용하는 대표적인 저작도구들이다.

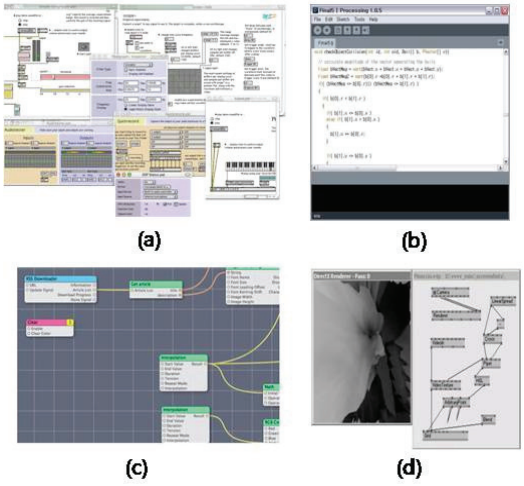


그림 2. 다양한 형태의 미디어 아트 콘텐츠 저작도구
(a) Max/MSP/Jitter (b) Processing
(c) Openframeworks (d) VVVV

Max/MSP/Jitter는 1986년 프랑스 IRCAM에서 전자음악 디바이스를 컨트롤하기 위해 개발한 프로그래밍 언어이다. 이후 기능을 확장하고 플러그인 형식의 모듈을 외부 개발사나 사용자들이 개발하여 그 기능을 확장시킴으로써 현재까지 프로그램의 활용가능 영역을 넓혀왔다.

*Processing*은, MIT Medialab에서 프로그래밍의 어려움을 겪는 미디어 아티스트들과 디자이너들을 위해

쉽게 프로그래밍하고 실시간으로 그 결과를 확인할 수 있도록 개발한 미디어 아트 콘텐츠 저작도구로서, 2001년부터 공개적으로 개발되어 왔고, 그 후 발전을 거쳐 2008년에 비로소 1.0 정식 버전을 발표하였다. 텍스트 기반의 프로그래밍을 통한 콘텐츠 제작이 가능하다.

Openframework는 C++를 기반으로 개발된 그래픽 라이브러리로, Parsons와 MediaLab Madrid, Hangar Center의 협력으로 Z. Lieberman과 T. Watson에 의해 개발되었다. *Processing*처럼 별도의 소프트웨어 제공은 없으며, 윈도우에선 비주얼 스튜디오나 맥에서는 Xcode[11]등 개발 SDK를 이용하여 사용할 수 있다.

VVVV는 Meso라는 독일 미디어 디자인 그룹회사에서 최초로 개발하여 그들만의 저작도구로 사용되고 있다가 일반인들에게 공개되었다. VVVV는 DirectX[12] 기반으로 만들어졌기 때문에 오직 Windows에서만 사용이 가능하다는 제약이 있다.

Adobe사의 Flash는 MX버전에서 카메라를 연결, 제어할 수 있는 기능을 제공하는 것을 시작으로 최근 CS4 버전에 이르러 각종 인터페이스 보드와 센서들을 활용할 수 있는 기능에 이르기 까지 미디어 아트 저작도구로서의 기능적인 영역을 확장하고 있다.

PureData와 Gem는 Max/MSP/Jitter와 동일한 제작자가 만든 공개형 저작도구 이다. PureData는 Max/MSP/Jitter를 사용하던 사람들이 저렴한 비용으로 음악 작업을 할 수 있다는 점에서 많은 뮤지션이나 아티스트들에게 각광을 받고 있다. 국내에는 아직 많은 유저가 없으나 디지털 음악을 공부한 국내 학생들 사이에서 고가의 Max/MSP/Jitter의 좋은 대안으로 여겨지고 있다. Gem은 Max/MSP/Jitter 프로그램의 Jitter와 같은 역할을 하는 것으로 그래픽 프로그래밍을 할 수 있도록 하는 프로그램이다.

Quartz Composer는 맥용으로 만든 미디어 아트 콘텐츠 저작도구이다. 그래픽 유저 인터페이스 방식을 채용하고 있으며, 사용하는 사람은 많은 편이나, 아직 개발이 부족하여 다양한 패치를 사용할 수 없다는 단점이 있다. 따라서 Cocoa[13]와 같은 맥 프로그래밍을 이용하여 자신만의 패치를 만들기도 한다. 이런 문제점 때문에 저작도구에 익숙하지 않는 미디어 아티스트들이 사용하기 어렵다.

2. 국 · 내외 기술 현황

국외의 경우 [그림 2]와 같은 다양한 콘텐츠 저작도구를 활용한 미디어 아트 콘텐츠 창작 활동이 이루어지고 있다. 특히 Max/MSP/Jitter와 *Processing*은 전 세계적으로 미디어 아티스트들 뿐만 아니라 미디어 아티스트를 양성하는 교육기관에서도 많이 활용되고 있다.

Max/MSP/Jitter와 *Processing*의 경우 인터넷 상에 미디어 아티스트들을 위한 커뮤니티가 형성되어 있고, 해당 커뮤니티에서는 저작도구의 기능 확장을 위한 외부 라이브러리들이 활발하게 공개되고 있어 저작도구로서의 입지를 넓혀가고 있다.

그러나 국내의 경우 국외와 달리 개방형 범용 저작도구의 개발이 미미한 실정이며, 그 이유는 다음과 같다. 첫째, 국내에서는 미디어 아트 아티스트들의 저변이 넓지 않아 미디어 아트 저작도구가 교육기관이나 아티스트들에 의해 폭넓게 활용되고 있지 않으며, 관련된 정보를 얻기도 쉽지 않다. 둘째, 미디어 아트 저작도구를 이용하여 콘텐츠를 제작할 경우 아티스트 혼자만의 작업으로 이루어질 수 있는 것이 아니라 프로그래밍 전문가의 도움을 받아야만 콘텐츠를 제작할 수 있는 것이 현실이다. 마지막으로 아티스트들은 미디어 콘텐츠 제작과정에 필요한 하드웨어 제어를 위해 피지컬 컴퓨팅 분야까지 학습해야 하는 어려움이 있다.

III. Eclipse 기반 GMF 기법을 이용한 저작도구개발

1. EMF와 GMF

Eclipse에서 제공하는 EMF(Eclipse Modeling Framework)[6]는 구조화된 데이터 모델을 기반으로 하는 애플리케이션을 개발하기 위한 모델링 프레임워크이자 개발 도구이다. EMF는 XML 문서상에 기술되어 있는 데이터 모델에 대응되는 자바 클래스들을 생성하기 위한 런타임 라이브러리와 관련된 여러 가지 도구들을 제공한다. EMF에서 가장 중요한 것은 Ecore[6]라는

모델링 파일이며, 이 파일은 모든 모델 사이의 상호 관계를 정의 내린다.

GMF는 Eclipse상에서 EMF를 기반으로 그래픽 유저 인터페이스 기반의 어플리케이션을 개발할 수 있도록 개발에 필요한 기술적 요소와 런타임(runtime) 라이브러리를 제공하는 개발 도구이다. GMF는 기본적으로 그래픽 유저 인터페이스 기반의 어플리케이션을 개발할 수 있도록 기본적인 기능을 제공하기 때문에 개발자는 많은 노력 없이 쉽게 그래픽 유저 인터페이스기반의 어플리케이션을 개발할 수 있는 장점이 있다. 이러한 장점을 활용하여, ECAS도 GMF 기법을 활용하여 개발하였다.

GMF 기법을 이용하여 어플리케이션을 개발하기 위해서는 우선 도메인 모델(Domain model)[6]을 정의한다. 도메인 모델은 클래스 다이어그램이라고 하는데, 프로그램에서 사용하게 될 클래스들을 정의하고, 클래스들 간의 상호관계를 명시한다. 도메인 모델을 정의한 후 도구 정의 모델(Tool definition model)[6]과 그래픽 정의 모델(Graphical definition model)[6]을 정의한다. 도구 정의 모델에서는 프로그램상의 사용자 화면에 나타나는 기능들을 정의하고, 그래픽 정의 모델에서는 이 기능들에 대한 그래픽 적인 모양을 정의한다. 사용자 화면에 나타나는 아이콘 모양의 기능 버튼들이 여기에 해당한다. 세 개의 모델 정의가 끝나면, 마지막으로 세 개의 모델을 매핑(Mapping)시켜 매핑 모델(Mapping model)을 생성한다. 생성된 매핑모델과 RCP(Rich Client Platform)[14] 박스를 이용하여 최종의 소스코드를 생성한다. [그림 3]은 GMF의 dash 보드로 GMF를 이용한 일련의 과정을 보인 것이다.

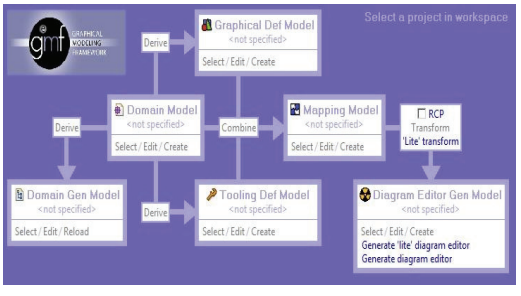


그림 3. GMF dash 보드

2. ECAS 개발 시 고려된 사항

ECAS는 다양한 기술적 요소를 활용한 미디어 아트 콘텐츠 제작을 목적으로 한다. 따라서 미디어 아트 콘텐츠 제작에 필요한 다양한 기술 요소를 제공해야한다. 예를 들어, Max/MSP/Jitter는 Max 부분이 비디오 처리 기능을, MSP 부분이 사운드 처리 기능을, 그리고 Jitter가 그래픽 처리 기능을 제공하여 미디어 아티스트가 하나의 저작도구 내에서 다양한 기술 요소를 조합하여 사용할 수 있다. *Processing*은 각 기술 요소들이 외부 라이브러리로 제공되기 때문에 미디어 아티스트들은 필요에 따라 라이브러리를 추가하여 해당 기능을 사용할 수 있다. 따라서 ECAS도 기존 저작 도구들이 제공하는 기본적이고, 공통적인 기능을 제공함으로써 저작도구로서의 범용성을 고려해야한다.

ECAS를 개발 시 고려되어야 할 또 다른 주요 고려사항은 기존의 저작도구에서 매우 세분화 되어 있는 기능을 통합하여 미디어 아티스트들이 사용하기 쉽게 기능들을 단순 명료하게 만드는 것이다. Max/MSP/Jitter의 경우 콘텐츠 제작에 필요한 기능들이 [그림 1(a)]와 같이 너무 세분화 되어 있어 저작도구에 익숙하지 않은 아티스트들이 사용하기에 어려움이 있다.

Max/MSP/Jitter의 경우 어떤 기능을 구현하기 위해 반드시 추가해야 하는 기능 블록들이 정해져 있다. 따라서 반드시 추가해줘야 하는 기능들을 하나로 묶어 통합하여 하나의 기능 블록으로 제공한다며 아티스트들이 세부 기능들을 하나하나 추가해야 하는 불편함이 없어 보다 쉽게 효율적으로 콘텐츠를 제작할 수 있다. 따라서 ECAS는 기존 저작도구에서 세분화 되어 있는 기능들을 통합하여 단순하고 명료하게 기능을 구현할 필요가 있다.

ECAS 개발 시 고려해야 할 또 다른 하나는 그래픽 유저 인터페이스이다. [그림 1(b)]에서 보듯이 *Processing*에서는 미디어 아티스트들이 콘텐츠를 창작하기 위해서 JAVA 기반의 프로그래밍을 직접 해야 한다. 그러나 프로그래밍에 익숙하지 않은 아티스트들에게 이 작업은 결코 쉬운 일이 아니다. 그러므로 ECAS는 이러한 어려움을 해결하기 위해, 미디어 아트 저작도구에 익숙하지 않은 아티스트들이 쉽고 빠르게 배워

사용할 수 있도록 그래픽 유저 인터페이스 방식을 기반으로 개발한다.

위의 내용들을 포함한 ECAS 개발 시 고려해야할 사항들을 정리하면 다음과 같다.

- 범용성(Generalization): 전문가 수준의 아티스트들뿐만 아니라 다양한 수준의 아티스트들이 쉽게 범용적으로 사용할 수 있어야 한다.
- 단순하고 명료한 통합된 기능(Simplicity): 복잡하고 세분화 되어 있는 요소 기능을 통합된 형태의 단순 명료하게 구현한다.
- 그래픽 유저 인터페이스(Graphic User Interface): 텍스트 기반의 프로그래밍 없이 그래픽화 된 기능 블록의 나열만으로 미디어 콘텐츠를 창조할 수 있다.
- 확장성(Extensibility): 향후 기능의 확장성을 고려하여 계속적으로 변화는 아티스트들의 요구에 대응할 수 있는 구조로 설계해야 한다.
- 외부 라이브러리 지원(Library): 외부 업체가 지원하는 라이브러리를 설치하여 해당 기능을 사용할 수 있어야 한다.
- 유지보수(Maintenance): 개발 후 업데이트를 포함한 유지 보수가 쉬워야 한다.
- 워크플로우(Workflow): 높은 수준의 미디어 아트 콘텐츠 제작을 위해 다양한 기능 블록들의 조합을 통한 워크플로우의 구성이 가능해야 한다.
- 사용자 정의 설정(User Configuration): 저작도구가 제공하는 각 기능에 대해 사용자 환경 설정이 가능해야 한다.

3. ECAS 기능 모델의 정의

ECAS는 미디어 아트 콘텐츠 제작에 필요한 여러 가지 기능을 가지고 있으며 각 기능은 JAVA기반의 클래스로 구현되어 있다. I.2절에서 언급했듯이, ECAS의 주요 기능은 크게 세 가지로 분류 할 수 있으며, 각 기능 분류는 ECAS 기능 모델링의 기본 클래스를 구성하고 있다. 예를 들어, *ImageProcessing* 클래스는 비디오 처리 기능을 담당하는 기반 클래스이며, *AudioProcessing* 클래스는 사운드 처리 기능을 담당하는 기반 클래스이다. 하드웨어 연동 처리 기능은 *Board*와 *MultiBoard*를 기반 클래스로 하고 이있다. 이 기반 클래스들은 *PBlock*이라는 최상위 클래스로부터 상속되어 구현되었으며, *PBlock*은 최상위 클래스로서 ECAS가 가지고 있는 모든 클래스들의 기반이 되는 클래스이다. 이 밖에도 미디어 콘텐츠 제작에 필요한 데이터 파일들의 입력을 담당하는 *InputSource* 클래스와 출력을 담당하는 *OutputSource* 클래스, 그리고 콘텐츠의 결과를 화면에 출력하기 위한 *Display* 클래스 등이 모델링 되어 있다. ECAS에서 사용하는 기능 블록에 대한 클래스 구성은 [표 1]과 같다.

표 1. ECAS 기능 모델의 클래스 구성

클래스 명	기능 설명
PBlock	최상위 클래스: ECAS 기능 블록의 기반 클래스
Connection	기능 블록들의 연결을 담당
ImageProcessing	이미지 처리 기능을 담당
AudioProcessing	사운드 처리 기능을 담당
VisualEditorDiagram	기능 블록의 워크플로우를 관리
Start	워크플로우의 시작점을 지정
OutputSource	데이터의 출력을 담당
InputSource	데이터의 입력을 담당
EventListener	프로그램 실행 시 이벤트 감지
ConditionControl	조건 판단이나 반복 기능을 담당
FaceDetect	ImageProcessing 클래스를 상속받은 클래스 :얼굴 검출 기능을 담당
VirtualOutput	OutputSource 클래스를 상속받은 클래스: 내부적으로 데이터 처리를 담당
OutputDevice	OutputSource 클래스를 상속받은 클래스: 최종적인 결과 출력을 담당
KeyEventListener	EventListener 클래스를 상속받은 클래스: 키보드 입력을 담당
MouseEventListener	EventListener 클래스를 상속받은 클래스: 마우스 입력을 담당
InputDevice	InputSource 클래스를 상속받은 클래스: 데이터 입력을 담당
VirtualInput	InputSource 클래스를 상속받은 클래스: 내부적으로 데이터 입력을 처리
SoundDetect	AudioProcessing 클래스를 상속받은 클래스: sound peak를 검출
If_block	ConditionControl 클래스를 상속받은 클래스: 조건문 판단 기능을 담당
SubEditorDiagam	하나의 기능 블록내에 다른 기능 블록을 포함 하는 기능을 담당
FileOutput	VirtualOutput 클래스를 상속받은 클래스: 데이터 파일을 저장하는 기능을 담당

Display	OutputDevice 클래스를 상속받은 클래스: 데이터를 화면에 출력하는 기능을 담당
Speaker	OutputDevice 클래스를 상속받은 클래스: 사운드 데이트를 출력하는 기능을 담당
Board	InputDevice 클래스를 상속받은 클래스: 외부 하드웨어 연동 기능을 담당
Camera	InputDevice 클래스를 상속받은 클래스: 카메라로부터 영상을 불러오는 기능을 담당
Microphone	InputDevice 클래스를 상속받은 클래스: 마이크로에서 사운드 데이터를 읽어오는 기능
EventGenerator	VirtualInput 클래스를 상속받은 클래스: 이벤트를 생성해주는 기능을 담당
FileInput	VirtualInput 클래스를 상속받은 클래스: 파일 데이터를 읽는 기능을 담당
MultiBoard	Board 클래스를 상속받은 클래스: 멀티센서 보드(H/W)와의 연동 기능을 담당
ArduinoBoard	Board 클래스를 상속받은 클래스: 외부 Arduino 보드와의 연동 기능을 담당
ToggleButton	EventGenerator 클래스를 상속받은 클래스: 0또 1 신호를 생성하는 기능을 담당
PushButton	EventGenerator 클래스를 상속받은 클래스: 버튼 클릭 이벤트를 처리하는 기능을 담당
CodeBox	FileInput 클래스를 상속받은 클래스로: 사용자 입력 코드를 처리하는 기능을 담당
ImageLoader	FileInput 클래스를 상속받은 클래스: 이미지 파일을 읽는 기능을 담당
AudioLoader	FileInput 클래스를 상속받은 클래스: 사운드 파일을 읽는 기능을 담당
Textviewer	VirtualOutput 클래스를 상속받은 클래스: 데이터의 전송이나 상태를 보여주는 기능
Serial	InputDevice 클래스를 상속받은 클래스: Serial 통신 기능을 담당
AudioEffect	Process 클래스를 상속받은 클래스: 오디의 실시간 효과 처리 기능을 담당
AudioMixer	AudioProcessing 클래스를 상속받은 클래스: 여러개의 사운드 데이터를 합성하는 기능

4. 구현

ECAS 구현은 크게 두 단계로 나누어진다. 첫째는 III.3절에서 언급한 기능 모델을 정의하는 것이고, 다른 하나는 해당 모델과 관련된 소스를 생성 시키는 것이다. 소스는 gmfgen[6]과 gmfmodel 파일[6]로부터 생성된다. 여기서 생성된 코드는 ECAS 소스코드의 기본 골격을 이루게 된다. [그림 4]에서 ECAS.beta6.diagram은 gmfgen 파일로부터 생성된 것이고, 나머지는 모두 genmodel 파일로부터 생성된 것이다.

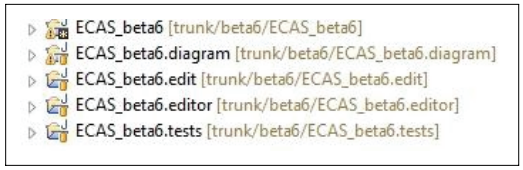


그림 4. ECAS 프로젝트 코드

생성된 5개의 코드 중에서 ECAS.beta6.diagram은 RCP 응용프로그램을 구현하는데 중요한 코드이며, 초기에는 제일 간단한 형태의 빈 클래스가 만들어진다. 여기에 해당 클래스의 기능에 맞는 코드를 구현하고 해야 하고, 이를 위해 두 가지 작업이 필요하다. 하나는 사용자에게 보이지 않는 내부 로직을 구현하는 것이고, 다른 하나는 그래픽 유저 인터페이스 연동을 통해 내부 기능을 사용자가 외부에서 기능 메뉴를 통해 사용할 수 있도록 하는 것이다. [그림 5]는 최종 구현이 완료된 ECAS의 사용자 화면이다.

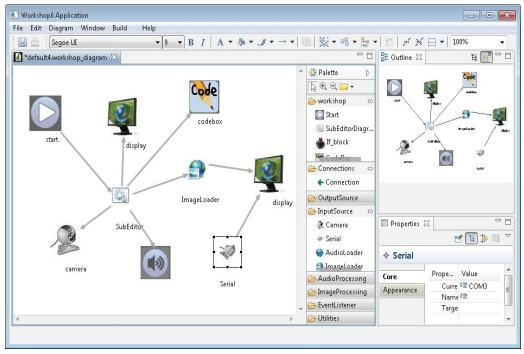


그림 5. 최종 구현된 ECAS의 사용자 화면

이밖에도 ECAS의 속도와 성능향상을 위해 몇 가지 방법들을 사용하였다.

- SWT(Standard Widget Toolkit)[15]를 이용한 그래픽 처리: SWT는 IBM에서 개발한 GUI 라이브러리로 직접 운영체제의 함수들을 호출하여 그래픽을 처리하기 때문에 속도가 빠르다.
- JIT(Just-In-Time)[16] 기반의 실시간 컴파일 기술: 프로그램이 실행되는 동안 JIT 컴파일러는 호출 빈도가 높은 함수를 기계어를 이용하여 직접 컴파일 한다. 따라서 JAVA코드를 기계어로 변화하

는 시간을 줄여 프로그램 수행 속도가 빠르다.

- 이미지 처리와 그래픽 처리를 위한 전용 라이브러리 사용: 이미지 처리와 그래픽 처리를 위해 JavaCV[17]와 Jogl[18] 같은 전용 라이브러리를 사용하였다. 두 라이브러리는 JAVA 언어의 JNI(Java Native Interface)[19] 방식으로, 기계어함수를 직접 호출하기 때문에 이미지와 그래픽 처리 속도가 빠르다.

IV. 실험 결과

본 절에서는 I.2절에서 언급한 ECAS의 세 가지 핵심 기능을 중심으로 기존의 저작도구와의 비교 구현을 통하여 ECAS의 편리성과 효율성을 보인다.

비디오 처리 기능은 웹 카메라와 같은 영상 입력 장치를 이용하여 실시간으로 영상을 입력 받고, 입력 받은 영상으로부터 미디어 아트 콘텐츠 제작에 필요한 정보를 추출하는 기능을 말한다. 본 실험에서 비디오 처리 기능 중에 하나인 얼굴 검출 기능을 이용하여 비디오 처리 기능을 테스트하였다.

ECAS를 이용하여 카메라로부터 입력되는 영상에서 얼굴을 검출하기 위해서는 [그림 6(a)]에서처럼 총 5개의 기능 블록이 사용된다. 전체 기능 블록들의 시작점을 지정하는 *Start* 블록, 카메라로부터 영상을 입력받기 위한 카메라 블록, 입력받은 영상에서 얼굴을 검출하기 위한 얼굴 검출 기능 블록, 검출된 얼굴을 영역을 표기하기 위한 영역 표시 기능 블록, 그리고 검출 결과를 화면에 출력하기 위한 화면 출력 기능 블록이 그것이다. 최종 실행된 얼굴 검출 결과 화면은 [그림 6(b)]와 같다.

얼굴 검출을 기존 저작도구 인 Max/MSP/Jitter나 *Processing*으로 구현하면 [그림 7]과 같다. [그림 7(a)]는 Max/MSP/Jitter를 이용하여 얼굴 검출 기능을 구현한 것이고, [그림 7(b)]는 *Processing*을 이용하여 구현한 것이다. [그림 7(a)]에서처럼 얼굴 검출 기능을 Max/MSP/Jitter를 이용하여 구현할 경우 Max/MSP/Jitter의 기능이 너무 세분화 되어 있기 때문에 많은 수의 기능 블록이 사용됨을 볼 수 있다. 그러나 [그림 6(a)]에서 보듯이 ECAS는 *Start* 블록을 포함한 5개의 기능 블록만으로 간단하게 얼굴 검출 기능을 구현할 수 있다.

[그림 7(b)]에서 보듯이 *Processing*의 경우 얼굴 검출 기능 구현을 위해 텍스트 기반의 프로그래밍이 필요하다. 프로그래밍에 익숙하지 않는 미디어 아티스트의 경우 얼굴 검출 기능을 이용하기 위해 [그림 7(b)]와 같은 긴 프로그램을 구현해야 한다는 점에서 미디어 아트 저작도구에 대해 상당한 부담감을 느낄 수 있다. 그러나 [그림 6(a)]의 ECAS의 경우 별도의 프로그래밍 기술이 없이 단지 몇 개의 기능 블록을 나열함으로써 얼굴 검출을 구현할 수 있다는 점에서 프로그래밍에 익숙하지 않는 미디어 아티스트들에게 편리함을 제공한다.

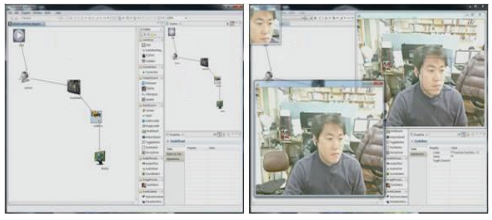


그림 6. 얼굴 검출 기능을 이용한 비디오 처리 테스트

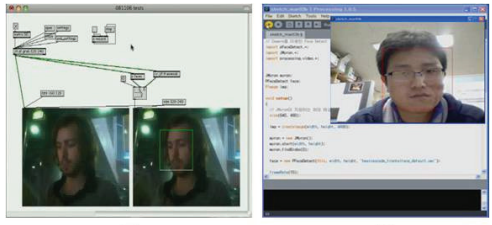
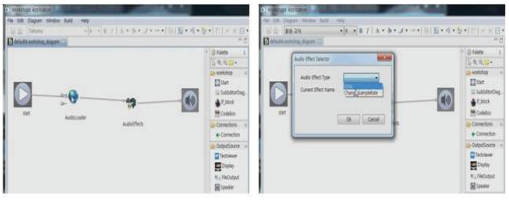


그림 7. 기존 저작도구를 이용한 얼굴 검출 기능 구현
(a) Max/MSP/Jitter (b) Processing

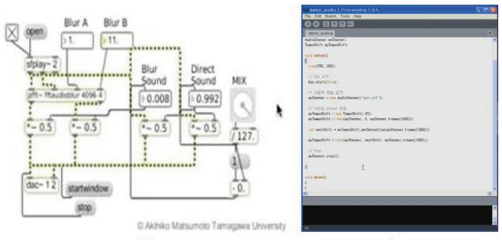
비디오 처리 기능뿐만 아니라 사운드 처리 기능에 있어서도 ECAS는 기존의 저작도구에 비해 많은 편리함을 제공한다. [그림 8]은 ECAS를 이용하여 사운드 효과 기능을 구현한 것이다. 사운드 효과 기능을 구현하기 위해 사운드를 입력 받는 것으로부터 효과를 적용하는 단계까지 [그림 8(a)]와 같이 단지 몇 개(여기서는 4

개의 기능블록)의 기능 블록을 나열함으로써 구현된다. 특히 사운드 효과 기능 블록의 경우 [그림 8(b)]처럼 미디어 아티스트가 원하는 사운드 효과를 기능 블록의 속성으로 설정할 수 있기 때문에 사운드 효과별 기능 블록을 별도로 추가하지 않아도 다양한 효과를 적용할 수 있는 편리함이 있다. [그림 9]는 사운드 효과 기능을 Max/MSP/Jitter와 *Processing*을 이용하여 구현한 것이다. [그림 9]에서 보듯이, 사운드 효과 기능을 기존 저작도구로 구현할 경우, 얼굴 검출 기능 구현과 마찬가지로 많은 수의 기능 블록이 필요하고 텍스트 기반의 긴 프로그래밍을 해야 한다는 불편함이 있다.



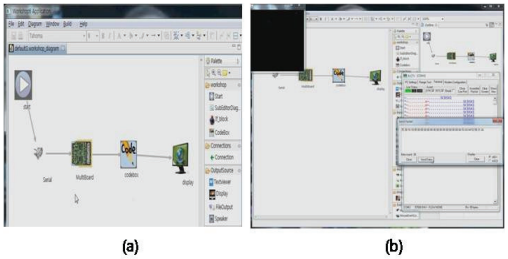
(a) (b)

그림 8. 사운드 효과 기능을 이용한 사운드 처리 테스트



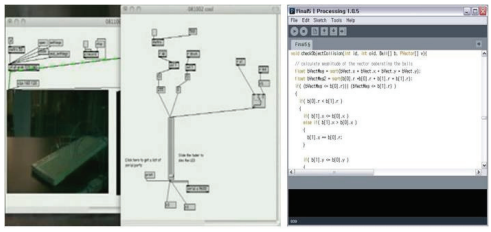
(a) (b)

그림 9. 기존 저작도구를 이용한 사운드 효과 구현
(a) Max/MSP/Jitter (b) Processing



(a) (b)

그림 10. 다중 센서보드를 이용한 하드웨어 연동 테스트



(a) (b)

그림 11. 기존 저작도구를 이용한 하드웨어 연동
(a) Max/MSP/Jitter (b) Processing

[그림 10]은 ECAS를 이용한 하드웨어 연동을 보인 것이다. 미디어 아트 콘텐츠 제작에 주로 사용되는 터치 센서를 ECAS에 연동하여 터치 센서의 값을 ECAS에서 사용하는 기능을 구현하였다.

하드웨어 연동의 경우, ECAS는 다른 기능들과 마찬가지로 [그림 10(a)]와 같이 5개의 기능 블록만으로 구현이 가능하다. 그러나 Max/MSP/Jitter는 [그림 11(a)]와 같이 기능 블록의 수가 더 많아지고 복잡해진다. *Processing*은 [그림 11(b)]와 같이 작성해야 하는 프로그램의 양이 늘어나게 되며, 이 경우 미디어 아티스트들의 작업 부담은 더욱 늘어나게 된다.

비록 본 논문에서 제안하는 ECAS가 그래픽 사용자 인터페이스와 기능블록의 간소화를 통해 아티스트들이 쉽고 편리하게 사용할 수 있기는 하나, 반면 몇 가지 한계를 가지고 있다. 첫 번째로, ECAS는 그래픽 사용자 인터페이스를 기반으로 하기 때문에 미디어 아티스트들이 요구하는 모든 기능들을 다 만족시킬 수 없다. 왜냐하면, 아티스트들의 복잡한 콘텐츠 창작에 필요로 하는 모든 기능들을 ECAS에서 제공할 수 없기 때문이다. 둘째로, 현재 버전의 ECAS는 실행 단계에서 약간의 시간적 지연 오차를 가지고 있다. ECAS의 각 기능블록들은 독자적인 단일 프로세서로 이루어져있으며, 이전 기능블록으로부터 처리할 데이터를 입력받는 순간 기능이 동작한다. 따라서 이전 블록으로부터 처리 데이터를 전달받을 때 까지 대기하게 되는데 이때 시간적 지연 오차가 발생한다. 마지막으로 현재의 ECAS로는 3D 데이터 및 기술을 활용한 콘텐츠 제작할 수 없다. 이는 현재 ECAS 3D 데이터를 처리할 3D 엔진이 포함되어 있지 않기 때문이다. 따라서 향후 ECAS에 3D 엔진을 탑

재하여 VR(Virtual Reality)나 AR(Augmented Reality) 기술을 활용한 콘텐츠 창작이 가능하도록 지원할 예정이다.

V. 사용자 연구

1. ECAS 사용자 체험 목표 및 개요

ECAS의 평가를 위해, 미디어 아트에 관심이 있는 다양한 수준의 미디어 아티스트 그룹을 선정하고, 사용자 평가를 위한 프로토타입(prototype)을 마련하여 미디어 아트 콘텐츠 저작도구인 ECAS의 사용에 편의성, 적합성, 유용성 등을 검증하였다.

사용자 체험 평가는 2달간에 걸쳐 총 2회 실시하였으며, 첫 번째 사용자 평가는 약 5일간에 걸쳐 1차 평가 프로토타입을 가지고 중급, 초급 사용자를 대상으로 실시하였다. 미디어 아트에 관심이 있거나, 실제로 작업에 참여하는 미디어 아티스트를 중급, 초급 그룹으로 나누어서 진행하였다. 본 사용자 평가를 위해 별도의 평가 프로토타입을 개발하여 진행하였다. ECAS의 사용자 평가는 공동연구기관인 이화여자대학교와 공동으로 진행하였다. 또한 본 사용자 연구 결과는 이화여자대학교에서 작성한 사용자 체험 보고서[20]를 참고하였다.

2. 체험 기간 및 장소

- ECAS 사용자 체험은 2010년 7월 26일부터 7월 30일까지 5일에 걸쳐서 진행함
- 이화여자대학교 이화캠퍼스센터 ECC B130호, B131호에서 실시함.

3. 체험 대상 선정

ECAS의 사용자 체험을 위해 체험자 그룹을 크게 초급과 중급으로 나누었다. 각 그룹은 6명의 규모로 구성하였다. 체험자 그룹을 초급, 중급으로 나누기 위해 사용자의 경험과 정보를 물어보는 사전조사를 진행하였다. 초급자는 기존의 저작도구, 예를 들면 *Processing*, Max./MSP/Jitter와 같은 소프트웨어를 전혀 다루어본 경험이 없는 경우를 말하며, 중급자는 기존의 저작도구를 다루어본 적이 있으며, 실제로 1-2개의 작품을 만들어본 경험이 있는 아마추어 아티스트들을 총칭하였다.

4. 사용자 평가 방법

ECAS의 사용자 평가는 설문, 관찰, 그리고 F.G.I (Focus Group Interview)의 세 가지 방법으로 수행하였다. 설문은 체험자의 인적사항, 기존 저작도구의 사용 유무등 사용자의 사전지식을 파악하는데 이용된다. 또한, 설문지를 통해 얻어진 답변 데이터를 통해 참가자들의 패턴을 파악할 수 있다. 관찰을 통해서는 사용자의 직관적 사용 능력 및 만족, 사용 편의성 등을 점검할 수 있다. 본 체험의 관찰을 위해 체험자의 옆에 관찰자가 동석하였다. 이는 ECAS 사용 시 체험자가 실제로 겪는 어려움을 상세하고 구체적인 부분까지 찾아내기 위함이다. F.G.I는 체험자와의 면담을 통해 체험을 평가하는 방법으로, 이를 통해 체험자의 다양한 이야기를 들을 수 있고, 직간접적으로 새로운 기능이나 디자인 방향에 대한 아이디어 획득이 가능하다. 또한 개발자와 디자이너가 해결점을 찾지 못했던 부분들의 문제점을 찾아내고, 체험자가 원하는 개선점을 도출하여, 그것을 ECAS 개선에 효과적으로 반영할 수 있다. 결론적으로 본 사용자 평가에서는 설문, 관찰 및 F.G.I의 방법을 함께 사용하여 사용성 평가의 정량적 객관성을 높이며, ECAS 개발의 기능성 및 사용 편의성을 극대화 하였다.

5. 사용자 체험 평가 기록

ECAS에 대한 사용자 체험 평가 결과는 [표 2]와 같다. GUI 부분은 가장 많은 지적을 받은 부분이며, 사용자가 가장 직접적으로 불편을 느끼는 부분이므로, 그래픽 사용자 인터페이스를 기반으로 하는 모든 프로그램들이 중요하게 다루어서 개발해야 한다. 다음으로 기능 개선 부분은 향후 시간을 가지고 점진적으로 기능을 안정화 및 개선해 나가야 한다. 마지막으로 매뉴얼을 사용의 편의를 위해 반드시 제공 되어야 한다.

표 2. 사용자 체험 평가 기록

번 호	분 류	평가 내용
평가 기록 1	GUI 개선	연결선을 이용한 블록 연결 방법이 직관적이지 못함. 기능블록들이 유사한 기능별로 그룹화가 필요. 기능블록의 입출력 노드의 크기가 작아 블록 사이의 연결이 어려움. 기능블록의 입출력 노드가 고정되어 있지 않아 실행하기 어려움. Editor창에 놓인 기능블록들의 일괄 정렬 기능이 필요함. 속성창의 속성값 명칭이 직관적이지 못함.
평가 기록 2	기능 개선	일부 기능블록들이 안정적이지 못함. 기능블록의 연결 순서 및 상관관계에 대한 이해하기 어려움. 콘텐츠를 실행하고 정지하는 기능이 명확하지 않음. 콘텐츠가 실행상태인지 정지 상태인지를 알 수 없음. 보다 많은 기능을 수행해 보고 싶으나 기능이 부족하고 제한적임.
평가 기록 3	기타 개선	매뉴얼이 없어 전체적인 사용이 어려움.

표 3. ECAS와 기존 국외 저작도구 개발의 선행사례 비교

	Max/MSP/Jitter	Processing	ECAS 2.0
개발자	밀러 푸켓(초안) –IRCAM(France) 실제개발 – 1990 David Zicarelli–Cycling74	1) Casey Reas and Ben Fry (MIT Media Lab) 2) Processing 2.0 Team	한양대 4개의 랩 협동 작업
개발년도	1980년중반 시작(밀러 푸켓) 실제개발 – 1990년 이후 1997년 상용화 – Cycling74 2004 – Jitter개발 추가	2001 개발확장 – Interaction Design Institute Ivrea, Carnegie Mellon University, and the UCLA 2002 processing discourse	2009–ECAS 1.0 개발시작 1차년도 – 요소기술 개발
현재버전	5.1버전(2009.07) – GUI update	1.2.1 (2010.07.14)	2010 현재 ECAS 2.0 개발
가능 플랫폼	Windows XP, MAC OS X	Cross–Platform	Windows
개발목적	컴퓨터 음악을 위한 프로그램 개발 / 2003년 이후 Jitter 등의 영상기반 작업 가능해짐	Visual art/Design 작업을 위해 개발	저작도구의 국산화 미디어 아티스트들을 위한 쉽고 편리한 새로운 저작도구 개발
개발형태	오픈 소스 프로그래밍	상용화 프로그램 (컴퓨터 음악+영상 기반의 프로그래밍)	현재 비공개로 개발 향후 상용화 예정
개발특징	GUI/C기반/현재 C, C++, Java, JavaScript 확장 가능	오픈소스 – 다양한 확장 라이브러리(70개 이상/웹 기반의 그래픽 구현가능 / 적은 용량과 낮은 컴퓨터 사양에서 구동가능 / 강력한 discussion forum 현재 NYU, UCLA, Lincoln Public Schools 등에서 교육용으로 사용중	GUI 기반의 직관적 Interface 제공 초, 중급자를 대상으로 한 미디어 아트 콘텐츠 저작도구 콘텐츠 제작에 필요한 특징적 기술 요소 추가
개발지원	Cycling74 회사	Miami University, Oblong Industries and the Rockefeller Foundaion 등의 지속적인 지원을 받고 있음 / Cooper–Hewitt National Design Museum – 내셔날 디자인 트리엔날레에 Processing 포함시킴	한국콘텐츠진흥원
가격	$495 Jitter(영상)포함 5.0버전 $699 (학생 9개월 – $59)	무료	미정

6. 사용자 체험의 종합 평가 및 결과

- 자체적인 내부평가를 통해 현재 ECAS 개발이 [표 3]과 같이 기존의 국외 저작도구 개발의 선행사례들(*Processing*, Max/MSP/Jitter)과 비교하였을 때, 짧은 개발기간에도 불구하고, 빠른 속도로 개발되었음을 인정함.
- [표 3]에 보듯이, 기존 저작도구들이 일반적으로 개발기간이 10년(최고는 20년) 이상이 필요했다는 점을 감안하였을 때, 짧은 기간에도 불구하고 저작도구로써의 면모를 갖춘 ECAS는 지속적인 시간이 주어진다면 기술적인 면에 있어 충분한 경쟁력이 있다고 판단됨.
- 현재 국내에는 시도하지 않았던 미디어 아트 저작도구개발이라는 새로운 시도(현재까지 디지털 미디어기반의 작업을 하는 수많은 아티스트들이 국외의 저작도구를 사용하고 있음)를 통해 시장성, 상용화 면에서 가능성이 있다고 사료됨.
- 기존의 저작도구에 비해 직관적인 인터페이스와 사용성에 있어서 쉬운 접근이 가능하다는 장점을 부각시키면, 교육용, 미디어 입문 혹은 대중적 체험으로써의 미디어아트 제작을 하기위한 저작도구로서의 역할을 할 수 있을 것이라 판단됨.
- 반면, 아직 남아있는 ECAS의 인터페이스 디자인과 매뉴얼 작성 및 사용성 평가에서 발견된 몇 가지의 문제점들을 사용자 체험 결과 보고서를 통해 수정 개선할 필요가 있음.

개발과정에서 발생하는 문제점들을 보완하기 위해 사용자 체험 및 미디어 아티스트들과의 협업을 지속적으로 할 필요가 있음.

VI. 결 론

본 논문에서는 Eclipse 기반 GMF 기법을 이용한 미디어 아트 저작도구인 ECAS와 그 개발 방법을 소개하였다. ECAS는 기존의 저작도구가 가진 텍스트 기반의 프로그래밍 방법을 개선하기 위해 그래픽 사용자 인터페이스를 기반으로 개발되었으며, 이를 통해 미디어 아티스트들이 보다 직관적으로 콘텐츠 창작 작업을 할 수 있게 되었다. 그래픽 사용자 인터페이스의 효율적인 구현을 위해 Eclipse가 제공하는 GMF 기법을 사용하였다. 또한 너무 세분화 되어 있어 사용하기 어려웠던 기존의 저작도구의 기능들을 통합하여 새로운 형태의 기능으로 구현함으로써, 사용이 쉽고 편리하도록 개발하였다. ECAS의 속도 및 성능 향상을 위해 SWT, JIT등의 속도 향상 기법을 활용하였다. 얼굴 검출 기능, 사운드 효과 기능, 하드웨어 연동 기능의 관점에서 기존의 저작도구와의 비교를 통해 ECAS의 편리성과 효율성을 보였다. 따라서 ECAS를 통해 미디어 아트를 콘텐츠를 보다 쉽게 창작할 수 있고, 미디어 아트에 대한 접근성을 쉽게 하여 더 나아가 미디어 아티스트들의 많은 활동과 작품의 발전이 이루어질 수 있을 것이다.

참 고 문 헌

[1] 심한수, “인터랙티브 미디어 아트를 위한 기술 동향,” 계원논총, 제10집, pp.343-358, 2004

[2] http://cycling74.com/

[3] http://www.openframeworks.cc/

[4] http://vvvv.org/

[5] http://www.processing.org

[6] F. Budinsky, Eclipse Modeling Framework: A Developer's Guide, Addison Wesley, 2003

[7] 금영욱, 조정길, IT Energy Eclipse를 이용한 JAVA Programming, 아이디 Media, 2009

[8] R. Robert, Adobe Flash CS3 Professional Bible, John Willey & Sons Inc., 2007

[9] http/puredata.org/

[10] E. Sadun, Getting Started with Quartz Composer, O'reilly Media Inc., 2006

[11] J. Bucanek, Professional Xcode, John Wiley & Sons Inc., 2010

[12] M. Tom, Managed DirectX 9, Macmillan Computer Pub., 2003

[13] T. Erick, Cocoa Programming for MAC OS X for Dummies, John Willey & Sons Inc., 2009
[14] J. McAffer and J. Lemieux, Eclipse Rich Client Platform, Addision-Wesiey Pub., 2005
[15] L. Mihalkovic and M. Scarpino, SWT/JFACE in Action, Oreilly & Associates Inc., 2004
[16] H. Hiroyuki, JIT Implementation Manual, CRC Pr. 2009
[17] http://javacv.sourceforge.net/
[18] http://jogl.dev.java.net/
[19] S. Liang, The Jave Native Interface: Programmer's Guide and Specification, Addison-Wesley Longman Inc., 1999
[20] 이승아 외 6명, CT기반 미디어 아트 전시를 위한 저작도구 개발의 기술 검증을 위한 사용자 실험 평가 및 결과 보고서, 이화여자대학교 탠저블&모바일 미디어랩, 2010

저 자 소 개

곽 재 호(Jae-Ho Kwak) 정회원

- 1999년 2월 : 한양대학교 전자전자통신전파공학군(공학사)
- 2001년 2월 : 한양대학교 전자공학과(공학석사)
- 2007년 9월 ~ 현재 : 한양대학교 전자컴퓨터통신공학과 박사과정

<관심분야> : 영상처리, 영상보안, 패턴인식, 내용기반 멀티미디어 분석

박 송 림(Songlin Piao) 정회원

- 2008년 12월 : 상해교통대(공학사)
- 2011년 2월 : 한양대학교 전자컴퓨터통신공학과(공학석사)

<관심분야> : 영상처리, 소프트웨어 디자인

김 회 율(Whoi-Yul Kim) 정회원

- 1980년 2월 : 한양대학교 전자공학과(공학사)
- 1983년 2월 : Pensylvania Staet University 전기공학과(공학석사)
- 1989년 2월 : Purdue University 전기공학과(공학박사)
- 1989년 9월 ~ 1994년 2월 : University of Texas 조교수
- 1994년 ~ 현재 : 한양대학교 전자통신컴퓨터공학부 교수

<관심분야> : 영상처리, 컴퓨터비전, 패턴인식, 머신비전, MPEG-7