

Proceedings of

# SSCC

1st

Summer 2016



Proceedings of  
**SSCC 1st**

Web ver.

Proceedings of

Special Interest Group on Sub-Culture Conference 1st

# Proceedings of SSCC 1st



Proceedings of

**SS  
CC**

**SIG SC Conference 1st**

Summer 2016



# SSCC 1st

## SIG SC

Authors:

Akey, 블스, sapphire, KeV, 제이미, Otter, Chœ Byung-yoon, Mappy The Kat, 리아, 小栗戸栗

Editor in Chief:

Debugger

Cover Illustration:

아루

Design Cooperation:

Chœ Byung-yoon

## Proceedings of SSCC 1st Web ver.

**지은이** SIG SC Conference 1st Authors

**발행처** SIG SC 소재지 서울특별시, 대한민국

### 발행일

2016년 08월 21일 초판 1쇄

2017년 06월 21일 Web ver. 1판

Copyright © 2016-2017 SIG SC Conference 1st Authors.

Copyright for components of this work owned by others than SIG SC must be honored.

Proceedings of SSCC 1st에 수록된 각 저작물에 대한 저작권은 해당 저작물의 저자에게 있으며 SIG SC는 해당 저작물에 대한 비베타적이고 양도 불가능한 복제 및 출판권을 가집니다.

## SIG SC

SIG SC Special Interest Group on Sub-Culture 는 한국의 학술향 동인 서클로,

- [서브컬쳐에 관련된 학술적인 접근] 과
- [학문 분야들에 대한 서브컬쳐적 접근] 을 지향하고 있습니다.

2016년 봄 결성 이후 같은 해 여름, SIG SC Conference 1st를 개최하였습니다.

## SIG SC Conference 약칭 sscC

동인 서클 SIG SC에서 개최하는 컨퍼런스로  
참가자들이 서로의 활동 성과를 공유할 수 있는 자리입니다.

첫 컨퍼런스인 SIG SC Conference 1st가 개최된 아직까지,  
SIG SC Conference는 논리적 가상 컨퍼런스의 형태로 개최되고 있습니다.

## Proceedings of SSCC

여러분이 보고 계시는 바로 이 책!

Proceedings of SSCC는 SIG SC Conference의 내용을  
서적의 형태로 펴낸 결과물입니다.

# Editorial Preface

Proceedings of SSCC 1st를 펼쳐주신 여러분께, 먼저 환영의 인사를 올립니다.

SIG SC의 활동 분야가 다소 생소할 수도 있어, '학술향 동인 서클?' 하며 고개를 갸우뚱하셨을 분도 계셨을 줄 압니다만, 공부를 하시는 학생분들로부터 전문직 종사자 분들이나 고등교육과 연구에 함께 종사하시는 대학원생, 심지어 대학의 교수 등에 이르기까지 덕후는 어디에나 있고 어디에도 없으니까요. 의무 교육을 포함해 여러가지 교육을 받아오셨을 현대 국가의 시민에게는 학술이야말로 어쩌면 가장 친숙하게 덕질하실 수 있는 분야 중 하나가 아닐까 싶습니다. 서브컬쳐와 함께 벼무려진, 평소에도 늘 접하셨을 수도 있는 학술이라는 소재의 색다른 맛을 한 번 느껴보세요!

생물학에서 스토리 텔링, 전산학, 수학에 이르기까지 다양한 분야를 다룬 10개의 기사를 투고 받아 SIG SC Conference 1st에서 소개할 수 있었습니다. 개인적으로도 편집 과정에서 내용을 검토하면서 하나같이 흥미로운 기사뿐이라 생각했습니다. Proceedings of SSCC 1st를 읽어보시며 소개된 내용들에 대해 관심을 가지게 되시는 계기가 될 수 있었으면 하는 바람입니다.

다소 마이너한 분야인 학술향 동인 활동에 무모하게 뛰어들었습니다만, 함께 해 주시는 분들이 계셔서 서로의 성과를 공유하고 이렇게 출판물의 형태로 여러분들께도 소개해 드릴 수 있게 되었습니다. SIG SC Conference 1st의 참가자 여러분께 진심으로 감사드리며, 같이할 수 있어 영광이었다고 생각하고 있습니다. 또한, 코믹마켓 등에서의 사례를 소개해 주시고 학술 분야의 동인 활동에 대해 관심을 갖도록 만들어주신 '); DROP TABLE \*;님과 결과물을 서적으로 출판하는 데 필요한 여러 가지 정보를 소개해 주신 수심님께도 감사의 말씀을 드립니다.

끝으로 SSCC에 관심을 가져주신 독자 여러분께 감사의 말씀을 드리며 서문을 끝맺고자 합니다. **부디, 즐겨주시길!**

SIG SC, Proceedings of SSCC 1st 편집 대표자,  
*Debugger.*

## Table of Contents

SIG SC 소개	v
Editorial Preface	vi
Table of Contents	vii
<b>REVIEW</b>	
민달팽이의 교미 같이 농후한	1
- 생식세포를 이용한 생물학적 동성간 자손 생성의 현재와 미래	
<b>ARTICLE</b>	
한국 서브컬쳐 작품 속의 '한국적임'을 찾아서	6
- 레트로봇 작품의 사례를 중심으로	
<b>ARTICLE</b>	
아이돌마스터 신데렐라 걸즈 스타라이트 스테이지	13
이미지 리소스 분석하기	
<b>ARTICLE</b>	
Face Detection of Monoeye Character	17
<b>ARTICLE</b>	
CRF를 활용한 한글로 적힌	22
외래어 및 외국어 받아쓰기 인식	
<b>TUTORIAL</b>	
란코처럼 말하는 트위터 봇 만들기	27
- 파이썬으로 트위터 봇을 만들고, 자동으로 문장을 생성하기	
<b>TUTORIAL</b>	
Wrapping Low-level Graphics Library	33
for Interactive Application	
<b>COMIC</b>	
대신귀여운평셔널을드리겠습니다	41
<b>ESSAY</b>	
주토피아는 실재하는가	46
<b>ESSAY</b>	
○○위키가 알려주지 않는 수학 이야기	53
	59
	축전
	Authors
	ix



# 민달팽이의 교미 같이 농후한

생식세포를 이용한 생물학적 동성간 자손 생성의 현재와 미래

Akey

## I. 들어가며

“AxB<sup>1</sup> 결혼해라!”라는 발언이 삶의 한 부분으로 자리잡고 있는 가운데 해당 커플을 소재로 한 아이 만들기 2차 창작 활동 또한 증가하고 있다. 많은 경우 이들 부부가 생물학적으로 같은 성을 가지기 때문에 대부분의 창작물에는 판타지적 요소가 개입한다. ‘매드 사이언티스트 속성을 가진 등장인물이 데려왔다’나 ‘초자연적인 존재가 개입했다’가 대표적인 상황 설정이며, 이런 여건이 만들어지지 않는 작품에서는 막연하게 ‘자고 일어났더니 옆에 있었다’라는 서사를 통해 사건이 진행되기도 한다.

창작물에서 등장하는 생물학적 동성간

아이 만들기를 현대 과학으로 재현하는 것은 불가능하다. 가장 큰 이유는 시도라도 했다간 엄중한 법의 심판을 받기 때문인데, 생식세포를 꺼내어 배아를 만드는 행위를 비롯해 모든 인간대상연구는 일차적으로 기관위원회의 엄중한 심의를 받으므로 최소 100년 안에는 죽었다 깨어나도 이런 연구를 시작하게 해 줄 리 없다고 생각하자[2]. 생물학적 동성간 수정 가능성의 연구는 대부분 동물실험을 통해 진행되어 왔으며 현재까지 꾸준히 좋은 성과를 보여오고 있다. 본 Review Article에서는 연구의 중요성과 간략한 역사, 그리고 예상되는 미래에 대해 소개하며 이를 서브컬쳐 측면에서 분석해보고자 한다.

<sup>1</sup> 혹은 BxA. 커플링 연산자 x는 교환법칙이 성립하지 않는다

<sup>2</sup> 2개를 따로 세는 이유는 이들이 성별을 결정하기 때문이다. XX면 여성, XY면 남성

## II. 생물학적 동성 간 생식세포 결합의 한계

인간의 생식세포에는 22+1개의 염색체가 들어 있고 정보량은 0.5이다. 두 개의 생식세포, 일반적인 경우에서 정자와 난자가 만나면 염색체가 각각 짹을 이루어 총 22쌍 + 2개<sup>2</sup>의 염색체를 가지고 정보량은 1이다. 단순히 정보량이 0.5인 두 생식세포가 결합하는 것으로 완전한 개체를 만들 수 있다면 좋겠지만 임상 및 동물실험을 통해 밝혀진 사실은 부정적이다.

정자와 정자 간의 결합은 발생에 필요한 양분이 부족해 성립할 수 없기 때문에 제외하고, 빈 난자에 정자가 들어가는 것과 같이 우연한 기회로 정자에서 유래한 유전자만을 가진 수정란은 이상발달을 일으켜 Molecular Pregnancy라는 일종의 암으로 변한다[3]. 분열은 하는데 배아의 형태를 갖지 않고 마구잡이로 증식하여 통제 불가능한 세포 덩어리가 되어버리는 것이다. 모든 유전자가 난자에서 유래했을 경우 종에 따라 그 결과가 달라지는데, 많은 무척추동물과 일부 척추동물에서는 수정이 일어나 정상적인 수정란으로 발달 가능하며 이를 Parthenogenesis, 단성생식이라 부른다. 인간을 비롯해 일반적인 척추동물의 경우에는 유전자가 난자에서만 왔을 경우 정상적인 발달이 일어나지 않는데, 심장이 뛰는 정도까지는 발달하지만 그 이후로 더 이상 진행하지 않고 죽어버린다[4].

그러므로, 일단 법적 윤리적 문제는 넘어 간다 치더라도, 정자와 정자, 난자와 난자

간의 조합에서 각각 유전자만 빼내 수정시켜 키우려는 시도를 한다면 우리가 인간인 한은 일반적으로 실패할 수 밖에 없다. 어떻게든 과학적 방법으로 해결을 보려는 글리먹은 이과생들은 생물학적으로 동성인 최애 커플이 농후한 플레이를 해도 아이 같은 건 생길 수 없다는 사실에 현실을 부정하며 난폭해질 수도 있겠지만, 누군가 그랬지 않던가. 우리는 언제나 해답을 찾을 것이라고.

## III. 생식세포 개조하기: 패턴 변경

이 모든 사건의 배후에는 Genomic Imprinting(유전체 각인)이라는 특이한 현상이 자리잡고 있다. 멘델 유전은 기본적으로 부모에게서 온 유전자에 가중치가 없다고 가정했지만, 이후 연구에서 포유류의 유전자 중 최소 100개 이상에서 가중치가 있다는 것이 확인되었다. 유전자가 어머니에게서 왔을 때는 발현하지만 아버지에게서 왔을 때는 발현하지 않는<sup>3</sup> 이런 특이한 성질은 성별에 따른 DNA 메틸화로 인해 염색체가 풀리지 않고 뭉치기 때문에 발생한다. 쉽게 말해 유전자가 부모 중 누구에게서 왔느냐에 따라 주석 처리가 되어 있을 수도 있고, 아닐 수도 있다는 것이다. 생명체가 제대로 발달하기 위해서는 필요한 모든 유전자가 정상적으로 작동해야 하는데, 만약 이들이 전부 한쪽 성별에서만 전달되었다면 반드시 사용해야 하는데도 주석 처리가 되어 있어 사용불능이 되어 버리기 때

<sup>3</sup> 물론 역도 성립한다

문에 일반적으로 포유류에서는 생물학적 동성 간 자손 생성이 불가능하다고 말한다.

켜져야 할 유전자가 꺼져 있는 게 문제이니 그걸 다시 켜기만 하면 생물학적 동성이라도 아이만들기가 가능할 것이다. 주석 처리 패턴을 적절히 변경하여 유전자를 조절하면 정상적인 발달이 일어날 것이라는 발상을 가지고 실험한 결과, 2004년 난자만을 이용해 쥐 수정을 성공시켰고 성체까지 성장하는 것을 확인할 수 있었다[5]. 타겟이 되는 유전자는 Insulin-like growth factor 2(Igf2)와 H19로, 다른 유전자들에게 주석처리를 할 수 있는 잠재력을 가지고 있으면서 유전체 각인이 적용되는 대표적인 대상들이다. 정자에서는 Igf2가 켜져 있고 H19는 꺼져 있는 반면, 난자에서는 Igf2가 꺼져 있고 H19는 켜져 있다. 만약 우리가 난자에서 Igf2를 켜고 H19를 끈 다음 다른 난자와 수정시킨다면 원하는 목표를 이룰 수 있을 것이다. 즉 두 개의 마스터 스위치를 건드려 생식세포의 패턴을 수정하고자 하는 것이다.

실험에는 완전히 성숙한 난세포(fg: fully grown oocyte)와 미성숙한 난세포(ng: non-growing oocyte)를 사용하였다. 난자는 태어났을 때에는 미성숙한 상태로 있다가 개체가 성장하면 함께 성숙하며 패턴이 난자 고유의 것으로 바뀌기 때문에 완전히 자란 난세포는 가공에 적절하지 않다. 때문에 비교적 각인에서 자유로우며, 동시에 Igf2가 아직 켜져 있는 상태인 미성숙 난세포를 갖 태어난 쥐로부터 추출하여 사용하였다. 다음은 미성숙 난세포에서 H19를 끌 차례인데, 이 문제는 아주 단순하게

H19 유전자 일부를 삭제해버리는 것으로 해결했다. 유전자로부터 나오는 결과물이 고장나서 작동하지 않게 되므로 결국은 꺼버리는 것과 마찬가지의 결과를 얻을 수 있다.

가공시킨 미성숙 난세포와 일반 성숙 난세포를 인위적으로 수정시킨 결과 598개 샘플 중 371개가 포배기까지 발달했고, 이를 26마리 대리모에 착상시킨 결과 28개 샘플이 완전한 발달에 성공하여 18마리는 사산, 8마리는 2주 안에 모두 사망하였고 오직 2마리만이 살아남았다. 착상 시도한 샘플의 0.6%만이 생존한 셈이라 확률 자체는 매우 낮지만, 그 중 한 마리는 성체까지 성장해 새끼를 낳는 데에도 이상이 없음이 밝혀짐으로서 난자 간 결합으로 온전한 개체를 만드는 실험은 성공했다고 볼 수 있다. 새끼를 낳은 이 쥐는 “카구야”라는 이름을 가지게 되었는데, 그 어원은 대나무 안에서 태어났다는 카구야히메이다.

## IV. 그리고 문제는 계속된다

여기까지는 좋았는데, 아직 해결되지 않은 문제는 산더미이다. 일단 실험에 활용된 이론의 정확한 매커니즘을 모르는 관계로 실전에 활용하기엔 여러 가지 위험이 따른다. 생물학 연구는 노드 4만개짜리 네트워크 역공학과 같아, 정확히 어떤 노드끼리 연결되어 있는지, 신호 전달이 어떻게 될지에 대해 확신할 수가 없다. 즉 수없이 많은 후속 연구가 진행된 후에야 이 이슈를 임상에서 진행할 수 있다고 보는 것이 적절하다. 당장 성공률이 0.6%인 것만을 보더라도 아직 본격적으로 적용할 수 있는 수준

은 아니라고 본다.

다른 문제는 정자 간 수정이 가능한지 여부가 확실하지 않다는 것이다. 유전체각인 패턴 변경에 대한 기본적인 지식이 있는 한 이는 시간이 지나고 연구가 진행되면 해결 될 문제이나 현재 시점에서는 성공 여부부 터가 확실하지 않은 일이기 때문에 함부로 단정지을 수 없는 상황이다. 역시 많은 후 속 연구가 필요하다.

가장 큰 걸림돌은 법적 윤리적 문제이다. 인간을 대상으로 하여 단성생식배아를 착상, 유지 및 출산하는 행위는 생명윤리 및 안전에 관한 법률에 의해 금지되어 있다. 정자 간의 수정 또한 핵이 제거된 난자를 사용해야 하는데, 인간을 대상으로 한 이와 같은 연구는 같은 법에 의해 금지되어 있다. 필요성에 의해 일부 법률이 개정되거나 하지 않는 한, 현실에서는 인간을 대상으로 연구가 불가능하기 때문에 대다수 연구가 동물을 대상으로 진행되고 있다. 물론 이런 제한 때문에 인간 복제 음모론이 나오기도 하는 것이지만 사회적, 윤리적으로 이 이슈가 받아들여지기 전까지는 아무래도 조심스러울 수 밖에 없다.

## V. 서브컬쳐적 측면에서의 접근 및 이론의 활용 방안

각종 제한으로 인해 본 이론을 현실에서 는 사용할 수 없지만 픽션 속이라면 여러 가지 창의적인 활용이 가능할 것이다. 하드SF 관련 설정으로 사용하거나 Another Universe를 구축하는 등 기본적으로 이 이론은 세계관 창작의 베이스, 또는 작품 내부의 핵심적 도구로 활용될 것이다. 세계

관에서의 기본 요구조건은 전체 유전체 정보가 이미 해석되어있거나 거의 해석이 완료된 수준의 생명과학 및 생명공학 기술이며, 부가적으로 인공자궁을 비롯한 육성기술이 필요하다. 후자의 경우 대리모 활용이 일반적인 환경에서라면 반드시 필요한 요소는 아니다. 생명윤리는 현재와는 큰 폭으로 다르며 최소한 인간 배아 복제에 대해서는 용인되는 분위기가 보편적이어야 한다.

생물학적 여성의 경우 성염색체 조합이 XX이기 때문에, 이들 생식세포의 조합으로 발생하는 후손은 전부 여성이다. 이들 커플이 남자아이를 갖기 위해서는 추가적인 절차가 필요하며 이 과정에서 X염색체 하나를 제거해야 한다. 생물학적 남성의 경우 성염색체 조합이 XY이기 때문에 후손은 남성 혹은 여성일 수 있으며 Y 염색체가 부모 중 어느 쪽에서 오느냐의 여부를 결정할 수 있다. 이와 같은 선택의 문제가 때로는 취향의 문제로 번질 수 있으며, 큰 틀에서 생물학적 동성인 커플에서부터 나온 아이가 부모의 성별을 따라야 하는지부터 시작해 작게는 커플 중 누가 어떤 유전자를 제공하느냐의 문제로 리버스 논란이 벌어질 수도 있다.

## VI. 마치며

이상으로 생물학적 동성 간 생식세포 결합과 그로부터 비롯되는 아이 만들기 과정에 대한 과학적 서브컬쳐적 측면에서의 고찰을 진행하였다. SF 관련 설정덕질을 좋아하는 분들이 본 리뷰에 만족하셨으면 좋겠지만, 역시 손만 잡고 자고 일어났더니 황새가 아이를 물어다 주는 편이 좀 더 캐

주얼하다는 점을 부정하기는 힘들다. 아직 까지도 갈 길이 멀지만 언젠가는 우리가 원하는 커플링 모두를 현실에서 만날 수 있는 날이 올 것이다. 그 때가 되면 모두 함께 민달팽이의 교미와 같이 농후한 창작활동을 하도록 하자. sscC 1st

---

## References

- [1] S. F. Gilbert, *Developmental Biology* (Sinauer Associates, Sunderland, 2013), 10th ed.
- [2] 생명윤리 및 안전에 관한 법률
- [3] P. A. Jacobs *et al.*, *Nature* **286**, 714-716 (1980).
- [4] M. H. Kaufman *et al.*, *Dev Biol* **59**(1), 86-90 (1977).
- [5] T. Kono *et al.*, *Nature* **428**, 860-864 (2004).

# 한국 서브컬쳐 작품 속의 ‘한국적임’을 찾아서

## 레트로봇 작품의 사례를 중심으로

봄스

### I. 서론

서브컬쳐 속에 녹아있는 일본 문화에 대한 지적이 네티즌 사이에서 오간 바가 있다. 주로 학교를 배경으로 한 작품에서 묘사되는 학교 생활의 모습이 실제 한국 학생들의 모습과 괴리가 크다는 지적이 주된 내용이었다. 학생회가 교내에서 무척 큰 지위로 여겨지거나, 학교 축제를 위해 학급 단위로 분주히 무언가를 준비하거나, 점심 시간에 학교 옥상을 자유롭게 출입하거나, 입학식과 졸업식 때에 벚꽃이 휘날리는 등의 모습은 일반적으로 ‘한국의 학생들이 겪는’ 학창 시절의 풍경과는 거리가 있다. 이를 두고 일각에서는 ‘어차피 가상의 작품인데 이 정도 자유로움도 허가하지 못하느냐’며 상상력의 부재라 비난한 바 있지만 앞서 서술한 이러한 예시들이 실제와 거리가 멀다는 지적에서는 벗어날 수 없다.

이번에는 조금 다른 이야기를 들어보자. 어떤 작품에 ‘한국적’인 요소가 있다고 할 때 흔히 무엇을 연상하는가? 혹시 <김치 전사>(강영만, 2011) 같은 것을 떠올리지는 않았는가? ‘한국적’이라고 하는 말에서 자연스럽게 한복, 김치, 태권도 등의 키워드를 떠올리지 않았는가? 유감스럽게도 이른바 ‘한국적’이라고 알려진 컨텐츠들이 이러한 수준에서 머물고 있는 것이 현실이지만, 이와 동시에 ‘한국적임’이 무엇인지, 그리고 그것을 컨텐츠에 자연스럽게 녹이기 위해 우리는 얼마나 연구하고 고찰하고 있는지를 비판적으로 따져볼 필요가 있다. 조건반사적으로 ‘한국적’이라는 말에 태권도와 김치 같은 단편적인 키워드를 떠올리며 부정적인 반응을 내세우는 것으로 한국적 컨텐츠에 대한 이해를 다 했다고 생각하지는 않았는가? 비판과 심도 깊은 사고 없이 알레르기에 가까운 혐오반응만 내세

워서는 악순환을 돌 수 밖에 없다.

본 글에서는 작 중 '한국적임'을 작품의 큰 틀로서 잘 소화하고 있는 작품의 예시로 레트로봇 사의 <변신 자동차 또봇>(이하 <또봇>)과 <바이클론즈>를 소개하고 오늘날 국산 서브컬쳐 작품들이 가져야 할 '한국적임'에 대해 생각해보고자 한다. 본 글에서 '한국적임'은 '다수의 현대 한국인이 겪거나 심정적으로 이해할 수 있는 소재, 공간 등의 활용'으로 정의하였다.

## II. 본론

본격적인 문제 제기에 앞서 우선 '한국적임'의 필요에 대한 의문을 짚고 넘어가자. 현대 한국을 배경으로 하는 모든 창작물이 꼭 한국적일 필요가 있는가? 답은 절대로 그렇지 않다. 현대 한국을 배경으로 하고 있음에도 불구하고 오히려 한국적인 느낌을 최대한 배척하고 작품을 전개하는 것이 더 효과적인 경우도 많다. (서브컬쳐로 부르기는 힘들지만) 대표적으로 박찬욱 감독의 <친절한 금자씨>(2005)를 비롯해 여러 작품들이 이러한 기법으로 몽환적인 분위기를 잘 살려낸 작품이라고 볼 수 있다. 그렇다면 작품에서 국적성을 무조건 배제하는 것이 옳은가? 기법에 따라 선택적으로 소거할 수도 있겠지만 국적성을 가지는 것이 작품 내외에서 효과적으로 작용하는 경우 또한 많다.

국적성을 포함한 작품이 가지는 장점은 작품의 소비자에게 있어 작품의 이해와 몰입에 큰 도움을 줄 수 있으며, 그 분류는 크

게 두 가지로 생각해 볼 수 있다.

첫 번째는 공간적 몰입의 측면이다. 현대 도시를 배경으로 하는 어반 판타지나 일상물의 경우 창작물의 소비자와 작품이 가까울수록 소비자는 더 쉽게 작품에 몰입할 수 있다. 일본의 '라이트 노벨'로 지칭 되는 어반 판타지들이 이런 특성을 많이 가지고 있으며, 굳이 라이트 노벨이 아니더라도 일상의 공간을 작품으로 끌어오는 시도는 다양하게 존재하였다. <디지몬 어드벤처>(토에이 애니메이션, 1999)는 '선택받은 아이들'이 '디지털 월드'라는 이세계에서 모험을 하는 내용을 다루고 있지만, 이야기의 종반에서는 일본으로 돌아와 '후지 테레비' 건물에서 적과 대결을 펼친다!. 그 이후 후지 테레비 건물은 디지몬을 좋아하는 사람이라면 누구든지 한 번은 방문하는 장소가 되었다. 이처럼 현실의 공간과 작품의 공간이 가까운 경우 창작물은 보다 소비자에



**FIGURE 1.** 애니메이션 '디지몬 어드벤처'에서 등장한 후지테레비 건물(상)과 오다이바의 실제 후지테레비 건물(하). 디지몬 애니메이션 시리즈는 후지테레비를 통해 송출되었다. 이후 후지테레비 건물은 오다이바에서 유명한 관광 명소가 되었다[1].

<sup>1</sup> 국내 상영 시에는 KBS 본사 건물로 로컬라이징 되었다

게 친숙하게 다가가는 동시에, 소비자 입장에서도 현실과 좋아하는 작품을 걸쳐 볼 수 있다는 장점을 가진다.

두 번째는 정서적 몰입의 측면이다. 대표적인 예시로는 게임 <아날로그 : 어 헤이트 스토리>(크리스틴 러브, 2012, 이하 아날로그)가 있다. 시작부터 ‘남존여비’라는 키워드를 내세우는 이 작품 25세기의 미래를 다루고 있지만 ‘무궁화 호’라는 세대선 안의 상황은 조선 시대의 남존여비 사상에 의해 지배되고 있었다는 독특한 세계관을 선보였다. 제작자 크리스틴 러브는 주제를 전달하기 위해, 캐나다 태생임에도 불구하고 조선의 역사와 그 시대 여성들의 생활상을 각색하여 게임 아날로그를 만들었다. 작품을 통해 드러난 여성 차별을, 남존여비라는 말이 존재했던 조선의 역사나 풍습을 이해하고 있는 플레이어라면 더 빠르게 인지하고 주제의식에 쉽게 공감할 수 있다. 즉 작품에 드러난 국적성이 플레이어의 정서적 몰입을 쉽게 유도하고 작품의 배경 및 주제의식에 대한 이해를 돋는다는 점을 확인할 수 있다<sup>2</sup>.

그렇다면 현대 한국을 배경으로 한 작품은 어떤 식으로 한국적임을 자연스럽게 녹여낼 수 있는가에 대한 논의가 새롭게 떠오른다. 이 질문은 뒤집어 생각하면 ‘우리는 작품에서 무엇을 볼 때 한국적인 작품이라고 느끼는가?’ 하는 질문과 동일하다. 서브 컬쳐 작품을 향유하는 층에서는 이른바 ‘국뽕 작품’에 대한 피로를 호소하며 의도적으

로 한국적임에 대한 논의를 피하려는 경향이 나타난다고 해도 과언이 아니다. 이러한 고찰 없이 한국을 배경으로 한 작품을 만드는 경우 서론에서 언급한 것처럼 ‘분명 배경도 한국이고 등장인물도 한국인인데 웬지 읽다보면 마치 일본 작품을 읽고 있는 것 같은’ 웃지 못할 상황이 연출되기도 한다. 물론 이 글을 통해 ‘한국적인 것이란 이런 것이다’라고 딱 잘라 제시할 수는 없다. 앞서 정의한 것처럼 ‘한국적’이라고 느낄 수 있는 소재는 그 작품을 향유하는 사람이 경험이나 심정적으로 동의할 수 있는 범위로 제한되기 때문에, 소비자에 따라 서로 다른 견해를 가질 수 있기 때문이다. 따라서 여기서는 한국적인 요소를 작품에 잘 소화시킨 작품으로서 레트로봇의 또봇과 바이클론즈를 소개하고 각 소재가 어떻게 작품에서 몰입과 정서적 공감을 이끌어내는지에 대해 이야기 할 것이다.

한국 애니메이션 제작사인 레트로봇은 국내 시장을 겨냥하여 또봇을 제작하였고 큰 호응을 얻었다. 아이즈와의 인터뷰에서 레트로봇 대표 이달은 “<또봇>을 생활밀착형 애니메이션이라고 부른다. 개인적으로 면 나라 이야기나 판타스틱한, 혹은 다른 차원의 이야기는 별로 좋아하지 않는다. 좀 더 일상적이되 밀도 있게 벌어지는 이야기가 좋다. <또봇>에서도 우리 동네에 있을 법한 아이들이 파일럿이 된다. 그럴수록 시청자가 더 쉽게 몰입할 수 있지 않을까.” 라

<sup>2</sup> 영문 버전에서는 제작자의 말 항목에 참고한 조선사 관련 자료에 대한 설명이 적혀있는 반면, 한국어 버전에서는 ‘하지만 한국 플레이어들에게는 이것이 필요할 것 같지 않아서 대신 특별 메시지를 넣기로 했습니다!’라고 코멘트하고 있다.

며 밝힌 바 있다[2]. 그렇다면 레트로봇은 어떠한 시도를 통하여 ‘생활 밀착형 애니메이션’을 완성하였는가? 시청자의 입장에서 레트로봇의 작품에서 정의되는 ‘한국적임’은 의·식·주 소재를 활용한 것과 사회·정서적 소재를 활용한 것의 두 가지로 확인할 수 있었다.

즉 다양한 음식 모텔링을 선보이며 의·식·주 소재를 적극 활용하고 있다. 또봇은 10기~13기를 걸쳐 작품의 메인 소재로 분식집 프렌차이즈인 ‘어!김떡순’을 내세웠다. 여기서 레트로봇은 한식을 소재로 사용하면서도 위화감이 없을 정도의 자연스러운 스토리텔링을 선보임과 동시에 분식집이라

© 2014 레트로봇(주)



**FIGURE 2.** 바이클론즈 1기 1화 첫 장면은 올림픽 공원에 UFO가 착륙하는 장면으로 시작한다. 경기장 뒤로 펼쳐진 서울의 야경과 계단의 호돌이가 인상적이다[3].

의·식·주 소재의 활용은 작품에 비추어지는 소재 자체를 통해 시청자가 ‘한국적임’을 느낄 수 있는 소재들이다. 또봇과 바이클론즈는 의·식·주 소재를 통해 누가봐도 이곳은 배경이 한국이구나, 사건이 일어나고 이야기가 진행되는 곳이 우리가 사는 곳과 멀리 떨어져 있지 않구나 하는 인식을 시청자에게 심어준다. 또봇의 경우 해외로 수출하고 있어 직접적인 주소나 장소 묘사는 나타나지 않지만 사건의 배경이 되는 ‘대도시’는 명백히 서울을 모티브로 하고 있다. 또한 ‘먹방 애니’로 유명세를 떨칠만

는 이미지가 한국인에게 주는 그 친근함을 자연스럽게 시청자들에게 유도하였다.

한편 또봇보다 더 한국적 소재를 많이 활용한 바이클론즈는 직접적으로 지명이 언급되기도 하며<sup>3</sup>, 한강 둔치나 경복궁처럼 잘 알려진 지역을 배경으로 사용하기도 하였다. 이 소재들은 작품에 대해 세대를 아우르는 친밀감을 심어줌과 동시에 ‘알면 누구라도 웃을 수 있는’ 자연스러운 유머를 전달하고 있다. 한강 둔치를 달리는 썬캡 쓴 운동복 차림의 아줌마들을 엑스트라로 보여주거나, 심한 감기로 앓던 장남 오태오

<sup>3</sup> 바이클론즈 2기에서 밝혀진 주인공 오 남매의 집 주소는 ‘서울시 양구 애당동’이다

가 일어나서 가장 먼저 찾는 음식이 고춧가루 뿐인 콩나물 국인 것에서[4] 얻는 소재에 대한 공감력은 기획자의 의도 그 이상이라 단언할 수 있다.

기는 대도시에서 떨어진 어느 시골 마을 ‘널리리’를 배경으로 한다. 청년이 없고 노인과 어린 아이들로만 이루어진 이곳의 시골 풍경은 외국에서 보면 아주 낯설수도 있

© 2014 레트로봇(주)



**FIGURE 3.** 외계에서 온 우주선이라도 백미러에는 염주가 걸려있는 것이 상식이다. 진지하지만 동시에 제작진의 센스에 웃음을 금할 길이 없는 장면이다[5].

© 2014 레트로봇(주)



**FIGURE 4.** 학교 앞 문구점 (바이클론즈 3기). 자연스럽게 벽에 걸려있는 홀라후프와 봉지에 담겨 매달려있는 돼지저금통이 눈에 띈다. 유리 벽에는 ‘잉크 충전합니다’라는 글씨도 쓰여있다. 이보다 더 한국적인 배경이 있을 수 있는가?

나아가 레트로봇은 그 존재를 잠깐 비추는 것만으로도 시청자들에게 다양하게 공감이나 이해를 이끌어낼 수 있는 사회·정서적 소재를 적극 사용하고 있다. 또봇 18

지만 한국에서 볼 때는 전형적인 농촌 마을의 풍경이다. 마치 ‘여섯시 내고향’에서 볼 수 있을 법한 시골의 풍경을 그려내며 레트로봇은 자연스럽게 시청자들에게 있어 사

회적 현상에 대한 공감을 이끌어낸다.

한편 바이클론즈 3기에서는 주인공 오남매와, 더부살이를 하게 된 외계인 할머니 이순희가 윷놀이를 벌이는 장면이 나온

가 서로를 가족으로 받아들였다'라는 메시지를 전달한다. 한국적 소재를 통해 정서적인 공감을 간략하게 전달한 훌륭한 장면이라고 할 수 있다.



**FIGURE 5.** 윷놀이를 준비하는 풍경. 장판을 깔고 말판을 그리고. 편은 장기알을 골라 잡아 색으로 나눈다. 명절에 집에서 자주 볼 수 있는 풍경.

다. 배경 지식이 없이 보기에도 이 장면은 무척 어색할 수도 있다. 적 흄마 제국의 위협이 시시각각 닥쳐오는 상황에서 평화롭게 편을 갈라 윷놀이를 즐기고, 그 이후 이순희가 가족처럼 오남매를 아끼는 모습은 논리의 비약, 혹은 지나치게 많은 것을 설명하지 않고 넘어가는 것처럼 보일 수도 있다. 그러나 윷놀이를 문화적 맥락에서 해석하면 이 장면은 바이클론즈 3기 전체를 꿰뚫는 중요한 장면으로 바뀐다. 윷놀이는 평소에는 잘 하지 않는 전통 놀이이지만 명절이나 봄맞이 행사(최사대회)처럼 사람들이 모이는 곳에서 함께 어울려 노는 풍습으로 행해진다. 즉 바이클론즈 3기에서의 윷놀이는, 어울려서 윷놀이를 한다는 그 단순한 장면을 통해 시청자들에게 '함께 가족이 어울려 노는 놀이를 한다 = 오남매와 이순희

이 외에도 작품에서 표현되는 한국적인 소재는 다양하다. 또봇에서 주인공 하나, 두리 형제는 다문화 가정의 일수, 유진과 친구가 된다. 아버지의 사정으로 이사를 가야하는 독고 오공 형제가 간 곳은 어디 먼 타국이 아닌 거제도이다. 바이클론즈에서 장남 태오는 야식 배달로 생계를 유지한다. 악역 화심이 쓰레기로 이루어진 문명 종결 수를 키운 곳은 예전에는 난지도로 불리던 하늘공원이며, 같은 생각을 해서 뇌파를 하나로 모아야 합체할 수 있다는 말에 오남매가 떠올린 것은 집에서 끊여먹는 끈하고 얼큰한 부대찌개다. 또봇과 바이클론즈들이 있는 풍경을 떠올렸을 때 그곳이 먼 우주도 아니고 다른 차원도 아닌, 오늘의 대한민국이라는 것은 오늘날 작품을 보는 시청자들의 연령과 상황에 상관 없이 비슷

한 공감을 이끌어낼 수 있으며, 더불어 작품의 애청자들에게는 ‘아, 여기!’ 하고 작품의 한 장면을 현실에서 떠올리게 만드는 각별한 감흥을 선사하기도 한다. 또봇과 바이클론즈가 7세 이하 아동들을 겨냥한 작품임에도 불구하고 성인들까지도 두터운 팬층을 확보하고 있는 것이 이러한 소재의 차용들과 무관하리라고는 생각하지 않는다.

### III. 결론

지금까지 레트로봇의 작품을 중심으로 한 국적 소재의 활용에 대해 알아보았다. 작품에서 전달하고자 하는 내용이 명백하게 시청자들에게 전달됨에도 불구하고, 모든 작품에서 이와같은 소재를 차용하는 것은 현재의 서브컬쳐 시장에서는 분명한 한계를 가지고 있다. 팔리지 않는다는 것이 그 주요한 이유이다. 레트로봇에서 또봇을 만들 수 있

었던 것도 ‘해외 시장에 팔지 못해도 상관 없으니 우리나라에서라도 성공하는 콘텐츠를 만들자.’라는 마음가짐에서 출발한 것이었다<sup>4</sup>. 애초부터 무국적성을 의도했다면 모를까 작품을 팔기 위해 국적성을 강제로 없애야 하는 것은, 국적성을 분명히 드러내고 있음에도 불구하고 이른바 ‘잘 팔리는’ 미국이나 일본의 작품들에 견주어 보았을 때 슬픈 일이다. 그러나 그러한 한계에도 불구하고 국적성을 ‘국뽕’이 아닌, 촌스럽지 않은 방법으로 녹여내고자 하는 시도는 서브컬쳐 전반에 있어서는 서사를 더 깊게 하는 양념으로 자리할 것이며, 향유 계층에 있어서도 더 큰 공감과 감동을 얻을 수 있는 기회라는 점을 인지하고 이를 연구, 발달시켜나가려는 자세가 필요하지 않을까 하는 감상으로 글을 맺음한다. SSCC 1st

### References

- [1] kenosatone, "디지몬에도 등장한 독특한 외관의 후지테레비 건물!," *Tourist Note JAPAN*. (25 Dec. 2015), <http://tourist-note.com/kr/sightseeing/20151225080043/>, Web. 15 Jul. 2016.
- [2] 위근우, "<또봇> | ② 우리는 <또봇>을 생활밀착형 애니메이션이라고 부른다," 아이즈. (20 May. 2014), <http://www.ize.co.kr/articleView.html?no=2014051613227273642>, Web. 15 Jul. 2016.
- [3] 이달 *et al.*, 바이클론즈 1회, 레트로봇 (よ), 2014.
- [4] 이달 *et al.*, 바이클론즈 시즌2 22회, 레트로봇 (よ), 2014.
- [5] 이달 *et al.*, 바이클론즈 3회, 레트로봇 (よ), 2014.

<sup>4</sup> “아마 다른 애니메이션 종사자들도 우리처럼 로컬한 배경이나 동시대적인 유머 코드 등을 넣고 싶을 거라고 생각한다. 그런데 그렇게 하지 못한다. 앞서 말했듯 대부분의 국내 애니메이션은 유아용이고 거의 모두 글로벌 콘텐츠를 지향한다. 해외에 팔 생각으로 만드니 특정 문화나 배경, 개그를 적극적으로 활용하기 어렵다. 나 역시 과거 다른 회사에서 애니메이션을 만들 때 그런 답답함을 느꼈고. 그런데 <또봇>의 경우 영실업 사장님께서 <또봇>을 해외 시장에 팔지 못해도 상관없으니 우리나라에서라도 성공하는 콘텐츠를 만들자고 하셨다. 덕분에 제작자로서는 예전에 하지 못했던 걸 마음 놓고 할 수 있었다.” (이달, 2014) [2]

# 아이돌마스터 신데렐라 걸즈 스타라이트 스테이지

## 이미지 리소스 분석하기

sapphire

### I. Introduction

<아이돌마스터 신데렐라 걸즈 스타라이트 스테이지>(이하 데레스테)는 아이돌마스터 신데렐라 걸즈 IP Intellectual Property를 활용해 반다이 남코 엔터테이먼트에서 제작한 리듬게임으로 2015년 9월 3일에 구글 플레이스토어, 9월 10일에 애플 앱스토어에 릴리즈 되었다. 최근 출시된 다른 모바일 게임들과 마찬가지로 데레스테는 유니티를 사용하여 제작되었는데, 유니티 애셋 디컴파일러를 사용하면 리소스를 추출할 수 있다. 이 문서에서는 데레스테의 리소스 중 카드 이미지를 추출하는 법을 다룰 것이다.

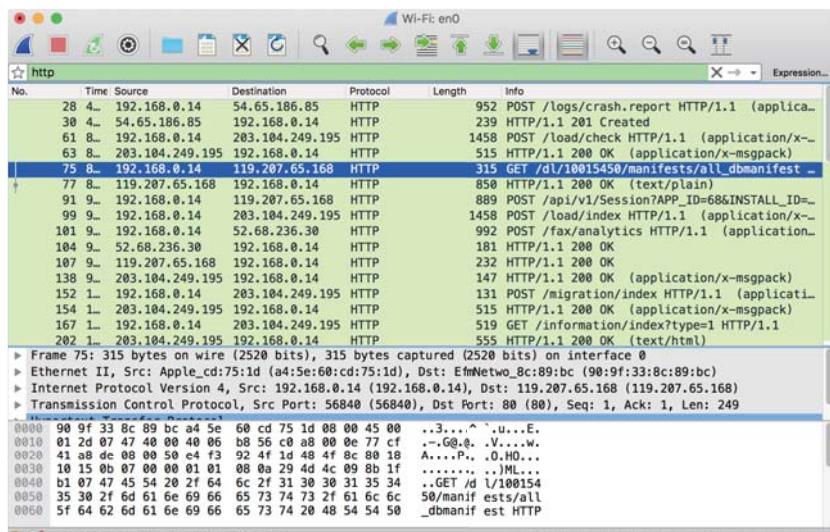


FIGURE 1. 패킷 캡처

## II. Downloading

앱스토어에서 데레스테를 설치하면 일부 리소스만 포함된 상태로 설치된다. 게임이 설치될 때 포함되지 않은 리소스는 외부 파일 서버에서 다운로드 받는데, 필수 리소스는 앱을 최초로 실행할 때, 나머지 리소스는 그 리소스가 필요할 때 다운로드 받는다.

WireShark와 같은 패킷 분석 툴로 데레스테의 HTTP 패킷을 스나핑하면 클라이언트에서 가장 먼저 하는 일은 /dl/{{res\_id}}/manifests/all\_dbmanifest에 요청을 보내는 것임을 알 수 있다({{res\_id}}는 리소스의 아이디로, 게임이 업데이트될 때마다 변경된다). 이 주소는 문자열을 반환하는데, 이 문자열에는 운영체제와 LOD에 따른 리소스 인덱스 DB의 파일 이름이 기록되어 있다. 리소스 인덱스 DB는 lz4로 압축된 sqlite3 DB 파일로 manifests라는 이름의 테이블에 개별 리소스의 원 파일 이름과 해싱된 파일 이름이 기록되어 있다.

	name	hash	attr	category	decrypt_key
1	3d_chara_body_...	e186b66e80ee24...	1	every	NULL
2	3d_chara_body_...	a31f53aa68c99...	1	every	NULL
3	3d_chara_body_...	fc7396c372d4dbf...	1	every	NULL
4	3d_chara_body_...	25b181cb29b090...	1	every	NULL
5	3d_chara_body_...	c08119b1e2d772...	1	every	NULL
6	3d_chara_body_...	0ddfd3cc9d28d8...	1	every	NULL
7	3d_chara_body_...	b1bad4b4ad8ae6...	1	every	NULL
8	3d_chara_body_...	3a7eed92b69486...	1	every	NULL
9	3d_chara_body_...	383a647b0cb5b1...	1	every	NULL

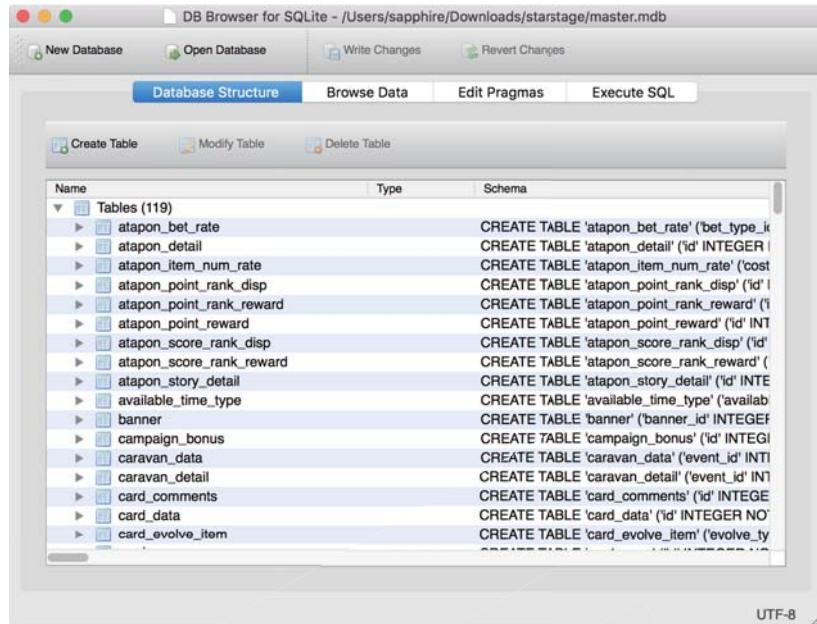
FIGURE 2. 리소스 인덱스 DB

## III. Unpacking

리소스 이미지를 추출하기 위해서는 이미지를 추출할 파일의 이름과 그에 해당하는 해싱된 파일 이름을 찾아 파일 서버에 요청을 보내야 한다. 그러기 위해서는 어떤 파일이 어떤 정보를 담고 있는지 알아야 하는데, 이 정보는 master.mdb라는 sqlite3 DB 파일에 담겨 있다. 리소스 인덱스 DB의 manifests 테이블에서 master.mdb를 검색하면 이 파일의 해싱된 이름을 구할 수 있다. 이 이름으로 서버에 요청을 보내면 리소스 인덱스 DB와 마찬가지로 lz4로 압축된 sqlite3 DB 파일이 다운로드 된다. 이 DB 파일에 있는 수많은 테이블

## 아이돌마스터 신데렐라 걸즈 스타라이트 스테이지 이미지 리소스 분석하기

에는 게임 내부에서 사용하는 정보들이 기록되어 있다.



**FIGURE 3.** master.mdb

파일 서버에서 특정 카드의 리소스를 다운로드 받으려면 그 카드의 고유 id를 알아야 하는데, 그 정보는 card\_data라는 이름의 테이블에 기록되어 있다. 카드의 고유 id는 6자리의 숫자로 큐트, 쿨, 패션 카드의 맨 앞자리 숫자는 각각 1, 2, 3이다. 위에서 얻은 카드의 고유 id를 리소스 인덱스 DB에서 검색하면, 그 카드에 해당하는 캐릭터 스프라이트, 아이콘, 배경 이미지 등을 찾을 수 있다.

1	id	name	chara_id	rarity	attribute	title_flg	evolution_id	series_id	pose	place	evolution
2	100001	島村卯月	101	3	1	0	100002	100001	1	1	
3	100002	島村卯月 +	101	4	1	0	0	100001	2	2	
4	100003	中野有香	102	1	1	0	100004	100003	1	1	
5	100004	中野有香 +	102	2	1	0	0	100003	2	2	
6	100005	持田里沙	107	1	1	0	100006	100005	1	1	
7	100006	持田里沙 +	107	2	1	0	0	100005	2	2	
8	100007	三村かな子	108	1	1	0	100008	100007	1	1	
9	100008	三村かな子 +	108	2	1	0	0	100007	2	2	
10	100009	奥山沙織	109	1	1	0	100010	100009	1	1	

**FIGURE 4.** card\_data 테이블

## IV. Decoding

데레스테의 이미지 파일은 JPEG나 PNG와 같은 형식으로 인코딩되어 있지 않고 자체적으로 만든 포맷을 사용하고 있다. 따라서 일반적인 뷰어를 사용해서 이미지를 보려면 이미지의 형식을 분석하여 다른 포맷으로 변환해야 한다. 이 문서에서는 이미지를 변환하는 방법은 다루지 않을 것이다.

캐릭터 이미지는 리소스 파일 하나당 한 장씩 들어있는데, `bg_{card_id}`는 카드 틀이나 글자가 포함되어 있지 않은 캐릭터 배경 이미지, `card_{card_id}_s`와 `card_{card_id}_m`은 카드 아이콘, `card_{card_id}_xl`은 카드 이미지를 가지고 있다. SSR 카드 한정으로 `gacha_{card_id}_{card_no}_sign`에는 캐릭터의 사인 이미지가 들어있다.



**FIGURE 5. card\_100252\_xl**

## V. Conclusion

데레스테는 유니티를 사용하여 제작되었기 때문에 상용 툴을 사용하여 애셋을 추출할 수 있다. 이미지는 자체 포맷을 사용하여 인코딩 되어있지만 압축 없이 픽셀의 RGB 값을 그대로 기록했기 때문에 분석하기 쉽다. 데레스테 리소스를 분석하면 이미지 뿐만 아니라 캐릭터의 3D 모델이나 2D 스프라이트, 애니메이션 데이터, 사운드 데이터, 텍스트로 저장된 캐릭터 정보 등도 추출 할 수 있다. SSCC 1st

# Face Detection of Monoeye Character

KeV

In this paper, we designed new face detection method for monoeye character, under assumption that characters has similar skin color and sclera color with human. Since monoeye characters are expressed to have large eye, our method focused to iris detection for face detection. Our method showed 17 true positives, 44 false positives, and 22 false negatives from 34 samples which contains 39 faces total, and showed better  $F_1$ -measure than related researches.

## I. Introduction

Face detection is image processing technique which detects face of human or character in image of movies. There are many methods to detect faces of human and to detect faces of characters, but there are no any methods to detect face of monoeye characters. Since monoeye characters have only one eye, it is hard to use approaches which made for non monoeye character.

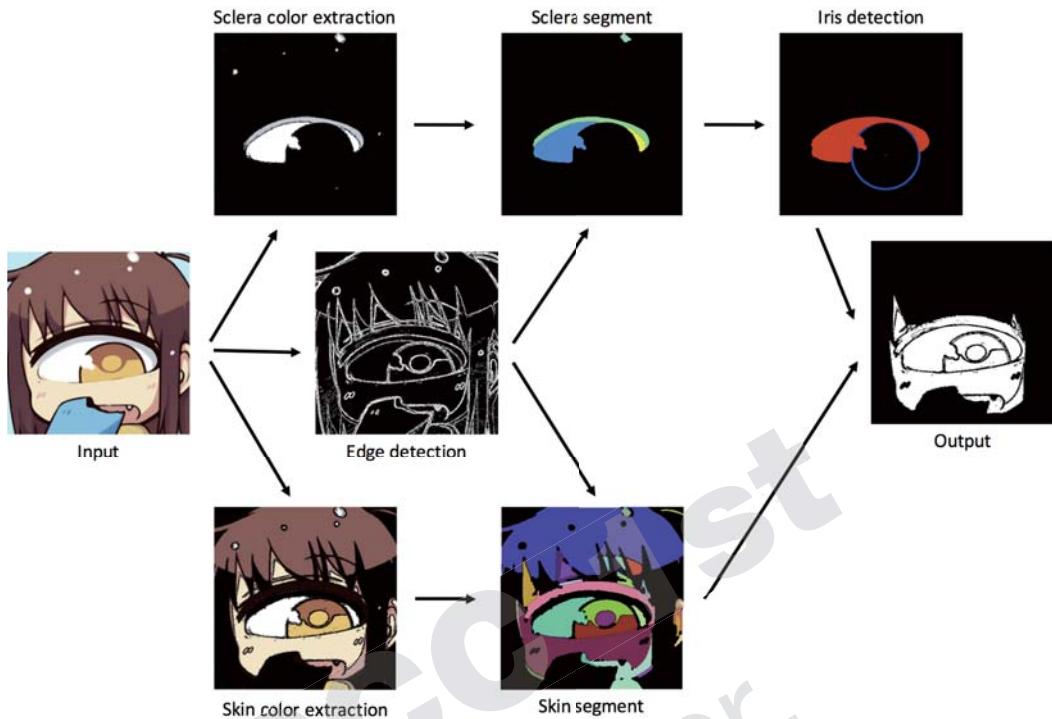
## II. Related Works

There are several methods to detect face of ordinary character. Face

detection method is often classified by two approaches; feature-based approach and image-based approach. Feature-based approach finds intrinsic features of faces, such as eye and mouth[1]. On the other hand, image-based approaches generates criteria of face from large data set, and uses generated criteria to detect face.

## III. Method

Since our main purpose is face detection of monoeye character, we should consider characteristics of monoeye character. Monoeye characters has only one eye, and



**FIGURE 1.** Flow of face detection of monoeye character

often expressed very large. So, our face detection method will use this characteristic to detect face. And also we will assume that skin and sclera color of character is similar with human, and skin and sclera color is not too similar. First step of our method is extraction of face color and sclera color, and edges of image. Second, our method will make segments of face and sclera. Third, our method will use circle regression to detect iris and eye. Last, our method will find face using detected eye and face regions.

### A. Edge Detection

Since making closed curve is very important to our method, our method used binarization after convolution with Laplacian operator. Our method uses CIELUV color space because it is perceptually uniform.

### B. Skin, Sclera Color Extraction

Jamie and Shaogaog described hue values of human skin colors are concentrated at 6 to 38 degrees[4], and Kohei, Henry and Tomoyuki arranged this condition to hue values in 0 to 40 degrees, and value over than 75%[1]. Our method will use this conditions

to extract skin color. Also, we will use equation 1-4 to detect sclera color[7].

$$\begin{cases} r = \frac{R}{R+G+B}, \\ g = \frac{G}{R+G+B}, \\ b = \frac{B}{R+G+B} \end{cases} \quad (1)$$

$$I_r = (r - 0.33)^2, I_g = (g - 0.33)^2 \quad (2)$$

$$I_c = I_r + I_g - 0.0009 < 0 \quad (3)$$

$$R + G + B > 1.2 * 255 \quad (4)$$

Where maximum value of R, G, B is 255. If pixel satisfies (3) and (4), our method recognizes the pixel as sclera colored pixel.

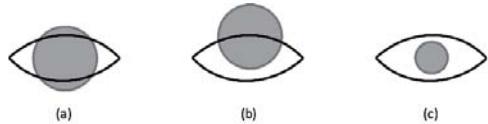
### C. Segmentation

Our method uses flood-fill approach to generate segment of skin color or sclera color. Skin and sclera segment is extracted from skin color pixel and sclera color pixel, and image edge. Segment is generated from skin or sclera color pixel, and grows until there is no any skin side pixel. And when segment is too small, segments are removed.

### D. Iris Detection

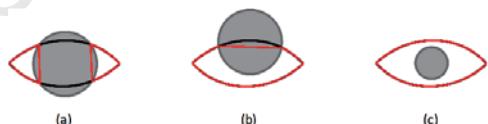
Our method detects eye from sclera segment to reduce false positive results. Since circle Hough transform is not effective for ellipse or smudged

circle, we designed new method to detect iris. Before to design method to detect iris, we should consider case of position between iris and sclera. There are 3 distinct case between iris and sclera.



**FIGURE 2.** Three distinct case between position of iris and sclera; (a) has two intersecting curves, (b) has only one intersecting curve, and (c) has no intersecting curves.

We can generate convex hull from using outermost contour of sclera segment, and the edge of convex hull will be calculated as following;



**FIGURE 3.** Edge of convex hull of outermost contour of Fig. 2

So, we can get the edge of iris partially from differential between edge of original segment of edge of convex hull. To detect iris from partially detected iris edge, we used RANSAC and circle regression which uses modified least squares method[5,6]. Also, our method did verification of iris using standard deviation of distance between detected

iris and each iris pixel. If standard deviation is larger than 20% of iris radius, detected iris is ignored.

## IV. Result

We used 34 distinct samples which contains total 39 monoeye characters. First, we analyzed results of our method step by step. The result is following:

	True Positive	False Positive	False Negative
Face Segmentation	36	-	3
Sclera Segmentation	31	-	8
Iris Detection	17	44	22

**TABLE 1.** Analyzed result of our method

Second, we compared face detection result of our method and other methods for non monoeye characters *TJN12*[1] and *Imager09*[3]. The result is following:

	True Positive	False Positive	False Negative	$F_1$ -measure
Our Method	17	44	22	0.34
TJN12	23	519	16	0.084
Imager09	8	9	31	0.039

**TABLE 2.** Result of face detection

## V. Discussion

Our method showed 27.8% of precision and 43.5% of recall, and 20.5% of false negatives are caused by sclera segmentation, 79.5% of false negatives are caused by iris detection. Since our method makes many false negatives at iris detection, we should improve our iris detection method

to improve performance. Also,  $F_1$ -measure of our method is larger than other methods, we can think our method is more efficient than other methods at detecting face of monoeye character. SSCC 1st

## References

- [1] K. Takayama, H. Johan, and T. Nishita, in *Proceedings of the IIEJ Image Electronics and Visual Computing Workshop 2012* 4B-1, edited by The Institute of Image Electronics Engineers of Japan (2012)
- [2] P. Viola and M. Jones, Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on 1, I-511 (2001)
- [3] "Imager::AnimeFace," *anime.udp.jp.*(2009), <http://anime.udp.jp/imager-animeface.html>, Web. 15 Jul. 2016.
- [4] J. Sherrah and S. Gong, "Skin Colour Analysis," *CVonline.* (18 May. 2001), [http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL\\_COPIES/GONG1/cvOnline-skinColourAnalysis.html](http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/GONG1/cvOnline-skinColourAnalysis.html), Web. 15 Jul. 2016.
- [5] M. A. Fischler and R. C. Bolles, Communications of the ACM **24**(6), 381-395 (1981)
- [6] D. Umbach and K. N. Jones, Instrumentation and Measurement, IEEE Transactions on **52**(6), 1881-1885 (2003)
- [7] A. Soetedjo, International Journal of Advanced Computer Science and Applications **3**(5), 17-22 (2012)

# CRF를 활용한 한글로 적힌 외래어 및 외국어 받아쓰기 인식

## 제이미

"하나무라로 떠납니다"라는 문장을 봤을 때 우린 "하나무라"가 일본 지명을 한글로 받아 적은 단어라는 것을 바로 알 수 있다. 비슷하게 "토도케테 세츠나사니와 나마에오 츠케요 오카 스노우 할레이션"을 봤을 때도 "스노우 할레이션"은 영어 단어를, 나머지는 일본어 문장을 그대로 받아 적은 한글이란 것을 바로 알 수 있다. 이 글에선 이러한 작업을 딥러닝을 활용하여 자동으로 인식 할 수 있게 해주는 방법을 제안한다. 주 목적은 한국어로 된 문장 속 외래어와 한국어 문장이 아닌 외국어 받아쓰기를 알아내는 것이다. 이는 추후에 개체명 인식이나 말뭉치 정제, 한본어와 같은 추가적인 외래어 분석 등에 활용 될 수 있을 것으로 보인다.

## I. 정의

### A. 외래어

외래어란 보통 외국에서 빌려서 한국어처럼 사용하는 단어 혹은 외국어의 고유 명사를 뜻한다[1]. 빌려왔다는 점에서 '차용어'라고도 한다. 전자의 예를 들자면 일 반명사 '텔레비전'의 경우, 그것을 지칭 할 한국어가 기존에 없었기 때문에 영어 'Television'에서 빌려와서 외래어 표기법으로 적은 후 한국어 단어처럼 사용한다. 후자의 경우 '러브라이브'가 대표적으로, 고유명사는 특정한 사람이나 기관, 작품 등을 구체적으로 지칭하기 위해 사용되기

때문에[2] 마찬가지로 외래어 표기법으로 적어서 사용한다.

### B. 외국어 받아쓰기

외국어 받아쓰기 dictation의 경우 외래어 와는 다르게 목적 자체가 의사소통이나 정보 전달에 있지 않고, 주로 외국어를 읽는데 도움을 줄 수 있도록 들리는 대로 한글로 적는 것을 뜻한다. 이를테면 "아루코오 하테나이미치"라는 구절은 노래를 쉽게 부를 수 있도록 일본어 가사를 한글로 적은 것이다.

## II. 알고리즘

### A. 품사 태깅

자연어처리(이하 NLP) 분야에서 가장 기본적으로 시행되는 프로세스는 품사 태깅 [3]이다. 각 단어별로 품사가 무엇인지 구분해서 '태그'를 달아두는 방식인데 이렇게 해두면 추후에 통계적인 방법을 통해 컴퓨터가 자동으로 여러가지 일을 처리 할 수 있게끔 해주는 일종의 방법론이다[4]. 가령 'lazenga save us'라는 문장이 있다면 'lazenga/명사', 'save/동사', 'us/대명사'처럼 각 단어의 품사를 기록해주는 것이다.

품사 태그를 은닉 마르코프 모델(이하 HMM)과 같은 방법으로 학습 하면 태그가 달리지 않은 단어에 대해서도 컴퓨터로 하여금 추론하게 할 수 있고[5], 여기서 명사만 뽑아서 주요 키워드로 선정한다거나 하는 응용이 가능해진다[7].

### B. 음절 단위 태깅

한국어의 경우 영어와는 달리 조사가 어미로 붙는 경우가 많기 때문에 단어가 띠어쓰기로 명확하게 구분되지 않는다. 따라서 대부분의 형태소 분석기는 한국어 맞춤법 규칙들을 활용해서 어느정도 사전에 분석을 해두고[6], 그 후에 HMM이나 조건부 랜덤 필드(이하 CRF) 등의 방법을 사용하여 품사를 유추해낸다. 그러나 딥러닝이 나오면서 최근 NLP의 트렌드는 언어학

적인 이론을 사용하지 않고 가능한 모든걸 학습으로 해결하는 방향으로 바뀌게 되었는데, 그 예로써 음절 단위로 끊어서 어미 규칙 등에 해당하는 부분까지 학습을 해보려는 시도가 등장하게 되었다[8,9]. 실제로 이 연구의 목적 중 하나는 한글로 적을 수 있는 모든 일본어 발음의 수가 백여가지를 넘는데, 그것들을 어떤 규칙으로써 미리 입력해두지 않아도 자동으로 알아내게끔 하는 것이다.

### C. 로케일 태깅

이 글에서 하고자하는 외래어 및 외국어 받아쓰기 인식의 경우, 각 음절의 발음이 중요한 특성feature이 될 것이라고 예상했기에 음절 단위 태깅을 사용했다. 태깅(혹은 레이블링) 문제에 있어서 탁월한 결과를 낸다고 알려진 CRF를 사용했으며, 실제로 동일한 방식으로 띠어쓰기를 자동으로 해주는 알고리즘이 연구된 적이 있다[10].

각 음절별로 '로케일 태그'라는 것을 달았는데, 이 로케일 태그는 ISO 639-1 코드[13]를 사용하며 가령 영어라면 en, 한국어는 ko, 일본어는 ja가 된다. 예를 들어 "우리 팀 한조 어디갔어"에 로케일 태그를 달게 되면 "우/ko 팀/en 한/ja 조/ja 어/ko 디/ko 갔/ko 어/ko"가 된다.

입력되는 문자열을  $x$ , 그 문자열의 로케일 태그들을  $y$ 라 할때 CRF는 다음 조건부 확률로 정의된다[10,11]:

$$p(y|x) = \frac{1}{Z(x)} \exp \left( \sum_j \sum_{i=1}^n \lambda_j t_j(y_{i-1}, y_i, x, i) + \sum_k \sum_{i=1}^n \mu_k s_k(y_i, x, i) \right)$$

그리고 실제 정답인  $y^*$ 에 대한 정확도가 높도록 각  $\lambda_j$ ,  $\mu_k$ 를 학습하게 된다. 위 식에서  $Z(x)$ 는 정규화 상수이며 전이 함수  $t_j$ 와 상태 함수  $s_k$ 는 직접 설정 할 수 있는 특성이다. CRF에 관한 자세한 설명은 [12]를 참고하기 바란다.

### III. 실험

#### A. CRF 설정

실제 구현은 python-crfsuite를 이용하였으며, 학습용 파라미터는 기본값을 사용했다. 특성으로는 현재 글자와 좌우 두 글자를 사용하였다. 실제로 실험에 쓰인 코드는 아래 페이지에서 확인할 수 있다[14]:

<https://github.com/theeluwin/sscc-1st>

여담으로, 이 코드는 굳이 로케일 태그가 아니더라도 그대로 적용할 수 있다. 가령 띠어쓰기를 할지 말지로 태그를 달았다면 [10]에서 소개한 띠어쓰기를 자동으로 해주는 알고리즘이 구현된다.

#### B. 말뭉치

'날개를 주세요' 등의 일본어 및 영어로 된 노래 가사를 발음대로 한글로 적은것과 나무위키의 '니시키노 마키' 항목과 같은 문

서를 활용하여 직접 ko, ja, en 태깅을 해서 말뭉치를 만들어 냈다. 수작업으로 했기 때문에 큰 편이 아니며, 딥러닝 특성상 학습 데이터가 많아진다면 더 좋은 결과를 낼 수 있을 것으로 예상되었기에 말뭉치의 크기를 변화시켜가며 실험했다. 학습 데이터는 크기를 각각 1배수, 2배수, 4배수 정도가 되게끔 설정했으며 (이하 학습-1, 학습-2, 학습-3), 로케일 태그의 비율은 약 en : ja : ko = 1 : 3 : 16 정도가 되도록 비슷하게 설정했다.

#### C. 결과

테스트 데이터에 있던 "생각해보니 츠바 사랑 마키 커플링이 없네"의 경우 "생/ko 각/ko 해/ko 보/ko 니/ja 츠/ja 바/ja 사/ja 랑/ja 마/ja 키/ja 커/en 플/en 링/en 이/ko 없/ko 네/ko"가 나왔다. 이는 '니코' 등에서 학습된 '니'가 ja로 인식된 것으로 보이며, '랑'의 경우 좌우 2글자씩이 전부 ja여서 그렇게 된 것으로 보이는데, 학습 데이터에 '랑'이 한국어 접속사로 쓰인 경우가 많지 않아서 학습하지 못한 것으로 예상된다. 반면 "토도케테 세츠나사니와"의 경우는 모두 ja로 잘 나왔다.

학습-3 데이터의 경우 세부적인 성능은 아래 표(Table 1)와 같다. 조건부 확률

로케일 태그	정밀도	재현율	$F_1$ 점수	테스트 글자수
en	0.88	0.83	0.86	2,296
ja	0.94	0.94	0.94	5,602
ko	0.98	0.98	0.98	37,257
평균/합	0.97	0.97	0.97	45,155

TABLE 1. 학습-3 데이터의 성능 평가

이 높은 순으로 여러개를 태그하여 정밀도(precision)와 재현율(recall), 그리고 다음과 같이 정의되는  $F_1$  점수를 계산해봤다.

$$F_1 = 2 \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

전체적인 정확도는 약 97% 정도가 되며, 학습-3 데이터에선 ko에 해당하는 글자가 총 77%나 되기 때문에 전부 다 ko로 태그 해주는 방법을 기준선으로 놓고 비교했을 때도 훨씬 더 좋은 성능을 나타난다고 할 수 있다.

그리고 아래는 학습 데이터의 크기를 변화시켜가면서 성능을 비교해본 표(Table 2)다.

말뭉치	전체 글자수	$F_1\text{-en}$	$F_1\text{-ja}$	$F_1\text{-kr}$	$F_1$ 평균
학습-1	24,470	0.78	0.89	0.97	0.95
학습-2	50,208	0.81	0.92	0.98	0.96
학습-3	100,936	0.86	0.94	0.98	0.97

TABLE 2. 학습 데이터 사이즈에 따른 성능 비교

확실히 학습 데이터의 크기가 커짐에 따라 전반적인 정확도가 올라가는 것을 볼 수 있다. 그리고 놀랍게도 학습-1 데이터의 경우, 전체 글자수가 테스트의 글자수보다 작은데도 상당히 높은 성능을 보였다.

## IV. 결론

### A. 의의

이 글에선 로케일 태깅을 CRF로 학습하여 한글 음절에 대해 자동으로 태깅 해주는 방법을 제시했다. 그동안 발전해온 한국어 형태소 분석기들 덕분에 한국어 NLP 처리도 많이 수월해졌지만 아직 외래어, 특히 외국어 받아쓰기들에 대해서 인식을 해내는 알고리즘이 없었으므로 새로운 제안이 된다. 외래어 인식은 개체명 인식(NER) 정확도를 올리는데 쓰일 수 있으며 외국어 받아쓰기의 경우 Word2Vec 등에 쓰일 말뭉치에서 제거하는 방식으로 활용이 가능하다. 한본어 분석이나 새로운 외래어를 알아내는 데에도 사용될 수 있을 것이다.

### B. 향후 연구 방향

CRF와 같은 딥러닝 알고리즘을 사용하지 않고 좀 더 고전적인 방법들이나 순수한 규칙 기반으로도 구현을 해서 기준선을 잡고 그것들과 정확도를 비교해보면 좋을 것이다. 또한 CRF의 특성을 어떻게 정의하느냐에 따라, 혹은 말뭉치에서 외래어와 외국어 받아쓰기 비율에 따라 성능이 어떻게 변하는지를 살펴보면 좀 더 높은 정확도를 지닌 알고리즘을 만들 수 있을 것이다. SSCC 1st

## References

- [1] 정희원, 새국어생활 **14**.2, 5-22 (2004)
- [2] 강경인, 영어영문학연구 **23**.1, 1-26 (1997)
- [3] K. W. Church, in *Proceedings of the second conference on Applied natural language processing*, (Association for Computational Linguistics, Stroudsburg, 1988), p. 136-143.
- [4] E. Charniak, AI magazine **18**.4, 33 (1997)
- [5] L. E. Baum and T. Petrie, The annals of mathematical statistics **37**.6, 1554-1563 (1966)
- [6] 심광섭 and 양재형, 정보과학회논문지: 소프트웨어 및 응용 **31**.1, 89-99 (2004)
- [7] 명재석, 이동주, and 이상구, 정보과학회논문지: 소프트웨어 및 응용 **35**.6, 392-403 (2008)
- [8] 심광섭, 인지과학 **22**.3, 327-345 (2011)
- [9] 권오옥, et al., in 1999 년도 제 11 회 한글 및 한국어 정보처리 학술대회 및 제 1 회 형태소 분석기 및 품사태깅 평가 워크숍 (한국정보과학회 언어공학연구회, 1999), p. 76-87.
- [10] 심광섭, 인지과학 **22**.2, 217-233 (2011)
- [11] J. Lafferty, A. McCallum, and F. C.N. Pereira, in *Proceedings of the eighteenth international conference on machine learning, ICML*. (2001), Vol. 1, p. 282-289.
- [12] C. Sutton and A. McCallum, arXiv:**1011.4088**
- [13] "ISO 639-2 Language Code List - Codes for the Representation of Names of Languages," *Library of Congress*. (18 Mar. 2014), [https://www.loc.gov/standards/iso639-2/php/code\\_list.php](https://www.loc.gov/standards/iso639-2/php/code_list.php), Web. 18 Jul. 2016.
- [14] theeluwin, "sscc-1st," *GitHub*. (18 Jul. 2016), <https://github.com/theeluwin/sscc-1st>, Web. 18 Jul. 2016.

# 란코처럼 말하는 트위터 봇 만들기

파이썬으로 트위터 봇을 만들고, 자동으로 문장을 생성하기

Otter

이 글은 독자가 파이썬의 문법을 간략하게 알고 있고, 트위터 API wrapper인 Tweepy에 대한 지식이 거의 없으며, 문장 자동 생성에 대한 기반 지식이 거의 없다는 가정하에 작성되었습니다.

이 글의 목표는 크게 두 단계로,

- 먼저 트위터 API의 wrapper인 Tweepy를 이용하여 트윗을 작성하는 것과,
- 작성할 간단한 문장을 생성해보는 것입니다.

설명은 기본적으로 파이썬 3버전대와 터미널 환경을 사용하는 것으로 가정하고 진행되지만, 2버전이나 다른 환경에서도 큰 문제는 없을 것으로 생각됩니다.

## I. Tweepy로 트윗을 하는 봇을 만들기

우리는 봇을 만들기 위해서 Tweepy <http://www.tweepy.org/>라는 파이썬 라이브러리를 사용할 것입니다. 먼저 라이브러리를 설치 할 필요가 있습니다.

pip를 사용하신다면

```
| pip install tweepy
```

를 터미널에 입력하면 설치가 가능합니다. virtualenv와 같은 가상 개발환경에 대한 설명이나 모듈의 다른 설치 방법에 대한 설명은 여기서는 생략합니다.

라이브러리가 설치 된 후에 트위터 봇의 뼈대가 될 코드를 작성합시다. 만약 파이썬 프로그래밍에 별로 익숙하지 않다면, 처음부터 코드파일을 작성하려고 하기보다는 python IDLE을 사용하시거나, 터미널에서 python 또는 python3 명령어로 REPL에 진입한 후 다음의 과정을 따라오는것이 편할 수 있습니다.

우리가 가장 먼저 해야 할 일은, 앞에서 설치한 라이브러리를 불러 오는 것일 겁니다. 이걸 하기 위한 첫 줄의 코드는

```
import tweepy
```

정도가 될 것 입니다.

다음 단계로, 트위터 앱을 등록해야 합니다. **Twitter Developers** <https://dev.twitter.com/>에서 Manage your apps 를 찾은 후 Create New App 버튼을 눌러서 나오는 항목을 작성하면 됩니다. 여기서 앱을 등록하는 개발자 트위터 계정과 봇을 운영하는 계정은 일치 할 필요가 없습니다.

앱을 등록하고 나면 나오는 리스트에서 사용할 앱을 누르면 다양한 설정이 가능합니다. 그중에서 일단 우리는 트윗을 작성해서 내보내는 봇을 만들고자 하기 때문에 Access Level을 설정 해 줄 필요가 있습니다. 이 옵션을 최소 Read and Write로 설정 해 줍시다.

그리고 Keys and Access Tokens 탭으로 들어가면 Consumer Key와 Consumer Secret이라는 것이 나옵니다. 이는 앱의 인증과정에 사용됩니다. 우리의 코드로 돌아가서, 다음과 같이 문자열 변수로 선언해 둡시다.

```
CK = 'App consumer key'  
CS = 'App secret key'
```

같은 식으로 작성하면 되겠습니다. 다만, 실제 코드로 작성할 때에는 조금 더 명시적인 변수명을(예를 들면, consumer\_key라던가) 이용해 주세요. 이 글에서는 분량의 제약상 저런 표현을 사용했습니다. 또한, Consumer Secret의 값은 공개되어서는 안 되니 주의할 필요가 있습니다.

이 키들은 트위터의 인증방식인 OAuth에 사용됩니다. OAuth가 무엇인지에 대한 자세한 설명은 이 글에서 다루지는 않겠습니다. 하여튼, 이 인증 방식을 위해서 tweepy에서는 OAuthHandler를 제공합니다.

```
auth = tweepy.OAuthHandler(CK, CS)
```

인증은 위와 같이 제공되는 핸들러로 처리를 하면 됩니다. 하지만 인증이 여기서 끝나진 않습니다. 다음 단계로

```
auth.get_authorization_url()
```

의 url을 열어서 로그인을 하고, 앱 사용 승인을 하고, PIN 코드를 발급받은 후에 이를 입력해줘야 합니다. url은 print 함수로 출력해서 직접 브라우저에서 열어도 되고, 파이선에 내장된 webbrowser 모듈을 사용하여 자동으로 해당 url을 기본 브라우저로 열게 할 수도 있습니다.

그 다음으로 이 PIN 코드를

```
verifier = input()
auth.get_access_token(verifier)
```

같은 식으로 터미널에서 입력 받게 하면 됩니다. input() 함수의 경우 파이썬 3버전대에서 사용하면 되고, 2버전대에서는 raw\_input()함수를 사용하면 됩니다.

마지막 단계로 얻은 액세스 토큰을 사용하여 인증을 끝마치면 됩니다.

```
AT = auth.access_token
ATS = auth.access_token_secret
auth.set_access_token(AT, ATS)
```

여기까지 하면 인증 절차는 끝이나고,

```
api = tweepy.API(auth)
```

와 같은 방식으로 이제 필요에 따라서 tweepy의 API를 호출 할 수 있습니다. 이런 인증 과정이 성가시게 느껴진다면, 이를 파일 등으로 저장해두는 것도 방법이 될 수 있습니다. 그리고 앱을 등록한 계정의 액세스 토큰 값 두 가지는 Keys and Access Tokens 탭에서도 확인이 가능합니다.

마지막으로 트윗을 해봅시다.

```
api.update_status('어둠에 삼켜져라!')
```

트윗 작성외의 다른 API에 대해서는

**Tweepy Documentation** <http://tweepy.readthedocs.io/en/> 을 참고하시길 바랍니다.

## II. 자동으로 문장 생성하기

사실 자동으로 제대로 된 문장을 생성한다는 것은 매우 어려운 일입니다. 특히, 한국어의 경우 다른 언어보다 연구가 덜 이루어진 분야이기도 합니다. 또한 비교적 초심자를 위하는 취지로 작성하는 이 글에서 깊은 배경지식을 요구하는 것은 적절하지 못합니다. 따라서 아주 간단한 두가지 방법론을 소개 하고자 합니다.

하지만 그보다 앞서서 먼저, 문법에 대한 이야기를 간단하게 하고자 합니다. 우리의 평소 발화나 작문이 문법을 철저하게 지키지는 않지만, 최소한의 문법도 지키지 않으면 문장이 부자연스러워 집니다. 간단한 예시를 들자면 이렇습니다.

란코의 말은 쿠마모토 사투리려나

말은 쿠마모토 란코의 사투리려나

분명 같은 단어들을 사용하더라도 순서의 차이만으로도 의미가 달라지거나, 또는 제대로 된 의미를 갖춘 문장에서 벗어납니다.

그렇다면 어떻게 적절한 문장을 만들 수 있을까요? 크게 보자면 두가지의 방법론이 있습니다.

- 문법적으로 옳은 구조를 모델링하고 그 모델을 사용해서 문장을 생성하기
- 예시 문장들을 확률론적으로 해석하여 임의의 문장을 생성하기

위의 두 방법의 가장 간단한 예시를 가지고 설명하겠습니다.

먼저 구조 모델링은 다음과 같이 이루어질 수 있습니다.

나는 학교에 간다

이 문장은 주어, 목적어, 동사의 어순을 가지고 있습니다. 우리는 이 규칙을 지키면서, 어휘를 바꿔넣는 방식으로 새로운 문장도 생성 할 수 있습니다. **나가 아닌 그, 학교가 아닌 수령, 가+ㄴ다 대신 빠지+ㄴ다**로 치환해 보면

**그는 수령에 빠진다**

라는 문장이 됩니다. 원본 문장의 문법적 형태는 유지하지만, 내용은 그와 무관한 문장을 새로 생성 할 수 있습니다. 단, 이렇게 생성된 문장은 문법적으로만 틀린 요소가 없을 뿐이지, 꼭 말이 된다는 보장은 없습니다. 같은 방식으로 생성 될 수 있는 문장에

**학교는 나무에 빠진다**

**소는 열매에 먹힌다**

같은 문장도 있는데, 이는 별로 자연스럽지 않게 느껴집니다.

예시 문장들의 확률론적 해석 중 간단한 방법은 다음과 같이 사용해 볼 수 있습니다.

**나는 아이돌이 되어서 빛나고 싶어**

**아이돌이 만만한 직업이라고 생각해?**

**만만한 일이라고 생각하지 않아**

라는 예시 문장들을 가지고 해석을 해보면, 우선 **아이돌**이라는 단어 뒤에는 **되어서**와 **만만한**이 오는 경우가 각각 한가지 있습니다. **만만한**이라는 단어의 뒤는 **직업이라고**와 **일이라고**가 오는 경우가 한가지씩 있습니다. **나는**의 뒤에는 **아이돌이**가, **일이라고** 다음은 **생각하지**, **생각하지** 다음에는 **않아**가 옵니다.

**나는**이라는 단어에서 시작해보면

**나는 아이돌이 되어서 빛나고 싶어**

**나는 아이돌이 만만한 직업이라고 생각해?**

**나는 아이돌이 만만한 일이라고 생각하지 않아**

와 같은 결과물이 나옵니다. 첫번째 문장은 우리가 예시로 제공한 문장과 동일하기 때문에 이상하지는 않지만, 두번째 문장은 앞과 뒤의 문맥이 달라서 자연스럽지 않습니다.

앞에서 소개한 두가지 모델의 장단점을 비교해보면 이렇습니다. 구조 모델링의 경우 확률론적 해석보다 문법적 규칙에 있어서 강력합니다. 확률론적 해석의 상대적인 장점은, 적어도 각각의 단어는 그 앞의 단어와는 유사한 맥락을 가진다는 점입니다. 구조 모델링의 경우 충분한 표본을 가지고, 형태소 분석과 병행하면 뼈대가 되는 문장 구조를 얻기에 좋아집니다. 확률론적 해석은 바로 앞의 단어에만 의존하지 않고, 앞의 n개 단어에 기반한다면 앞의 예시의 두번째 문장에서와 같이 갑자기 문맥이 달라지는 일을 다소 억제 할 수 있고, 역시 형태소 분석과 병행할 경우 동음다의어 때문에 생기는 문제를 어느정도 해결 할 수 있습니다.

그러면 우리가 문장 생성에 앞서서 트윗 작성은 하는 간단한 코드를 작성한 것처럼, 문장을 생성하는 간단하면서도 유의미한 구현체를 만들려면 어떻게 하면 좋을까요? 앞서 살펴보았듯이 문제는 크게 문법과 문맥 두 가지로 이루어집니다. 이 중 문법을 구조 모델링으로 상당 부분 해결하면, 문맥에만 집중할 수 있습니다. 그리고 일관적이고 한정된 주제와 맥락의 갖는 어휘들을 사용하면 문맥의 문제에도 도움이 됩니다. 이런 방식과 가장 가까운 실제 예시로는 주로 로봇 저널리즘이라고 불리는 알고리즘에 기반을 둔 뉴스 기사 생성이 있습니다.

그러면 우리는 기사 외의 어떤 것을 해볼 수 있을까요? 이 글의 제목으로 돌아가서, 저는 란코의 대사와 같은 문장을 만들고자 하였습니다. 다른 많은 가능성 중에 란코에 주목한 이유는 간단합니다. 평범한 문장을 템플릿으로 삼아서 새로운 문장을 만들어도, 중2병스러운 키워드 몇 개만으로도 나름의 개성이 드러나는 문장으로 바꿔게 됩니다. 그리고 어휘의 특성상 읽는 사람이 표면적인 의미를 가지고 말이 되지 않는다고 느끼는 것이 아니라, 다른 의미가 없나 나름의 추론과 해석을 시도하게 한다는데 의의가 있습니다. 요컨대 우리의 일상에보다 조금 더 문맥에서 자유로운 언어인 것입니다.

구조 모델링뿐만 아니라, 확률론적 해석도 좋은 결과물을 얻은 경우가 있습니다. 특유의 화법으로 유명한 **현 대한 민국 국가 원수** <http://markov.mathnt.net/>, **미국 차기 대선후보** <https://liph.github.io/markov/>에게 적용한 예시들은 정말 그럴싸해 보입니다.

추가로 조금 더 자세한 내용을 알고자 하는 분들에게 도움이 될 만한 자료나, 피드백 받은 내용 등을 <https://github.com/a19641sk/RankoBot>에 올려놓을 예정입니다.

마지막으로 존경하는 검사님, 이 글은 저희집 수달이 작성했습니다. sscC 1st

# Wrapping Low-level Graphics Library for Interactive Application

Chœ Byung-yoon

## 들어가며

이 글은 게임이나 암튼 그 3D 리얼타임 어플리케이션을 만들 때 꼭 필요하게 되는 부분인 그래픽스 렌더링 엔진 제작에 대해 다룹니다. 저수준 그래픽스 라이브러리를 어떻게 하면 잘 묶어서 좀 사람이 쓸만한 수준으로 만들까 하는 데에 있어 좀 도움을 줄 수 있으면 좋겠다는 생각에서 작성된 글입니다. 모쪼록 이 글을 읽고 그래픽스 렌더링 엔진을 만들 때의 고통이 줄거나, 책상 밑에서 대체 무슨 일이 일어나는지에 대한 이해를 함으로써 암튼 그 3D 리얼타임 어플리케이션 개발에 있어서 도움이 된다면 좋겠습니다. 우리도 인디게임 좀 많이 만들어봐야죠.

\* 이 글에서 다루는 저수준 그래픽스 라이브러리는 *OpenGL 3.1+*을 기준으로 기술됩니다.

## I. 서론

얏호- 모두들 안녕. 하는 건 없는데 고통은 받고 있는 대학원생 최병윤이라고 합니다. 앞서 써놨듯이 이 글은 저수준 그래픽스 라이브러리를 어떻게 하면 좀 사람이 쓸만하게 묶을까에 대한 글입니다 — 좀 더 자세히는 제시 가능한 디자인 초이스들과 왜 그런 선택을 했는지에 대해 설명하는 글입니다. 이 글의 내용 상당수는 KAIST 게임개발동아리 HAJE에서 진행된 프로젝트<sup>1</sup> 개발 중 발생한 렌더링 엔진 개발 경험을 기반으로 합니다. 각 항목들에 대해 얘기하면서 기본적인 설명은 확인을 목적으로 어쨌든 하겠지만 일단은 OpenGL에 대한 기본적인 이해를 하고 있다는 가정 하에 내용이 전개될 거예요. 난 삼각형도 어떻게 띠우는지 모르는데! 하시는 분들은 너무 패닉하지 마시고 다음 URL에서 제공하는 튜토리얼을 진행해 보시면 도움이 될지도 몰라요. <http://open.gl/>

<sup>1</sup> <https://bitbucket.org/haje/mmo>

혹시 읽다가 모르겠는 게 있거나 틀린 내용, 제안할 사항이 있다면 주저 없이 저자에게 메일 혹은 트위터로 연락 주세요.

## II. 버퍼 다루기

옛날 옛적엔 immediate mode라고 해서 뭐 그럴 때마다 GPU에 데이터를 넘겨주던 시절이 있었죠. 이는 곧 그 비효율성 때문인지 vertex의 정보들을 갖고 있는 buffer들로 대체됩니다. 덕분에 더 빨라지게 된 건 좋은데 어떻게 보면 굉장히 기본적인 동작에 조차 좀 더 러운 코드를 손에 물어야하게 된 원인이 되었죠. 어떻게 하면 이걸 좀 깔끔하게 감싸넣을 수 있을지 얘기해 봅시다.

### A. OpenGL에서의 Buffer Objects

요즘 GPU 아키텍처에는 CPU에서 직접 접근하는 RAM과는 별도의 온-칩 메모리가 달려 있죠. 하도 많은 데이터 액세스가 일어나다 보니 액세스 타임 좀 줄이려고 그렇게 설계를 해 놨는데, 그렇다 보니 여기 메모리도 또 따로 관리를 해 줄 필요가 생겼습니다. OpenGL에선 이걸 라이브러리에서 특정한 몇 개의 목적에 따라 컨트롤할 수 있도록 만들어 놨어요. 이것들을 Buffer Object라고 합니다. 각 buffer object들은 생성 generate될 때마다 목적별로 serial한 ID를 부여받게 되고, 얘네를 대충 C/C++ 프로그래밍에서의 포인터와 비슷한 느낌으로 써요. 생성도 했으니 당연히 제거 delete도 됩니다. 메모리 관리 열심히 해 줘야 돼요(눈물).

OpenGL은 기본적으로 state machine인데요, 그러다 보니 각 목적별로 buffer를 참조 할 일이 있을 때마다 마지막으로 bind()한 buffer object를 참조해요. 마지막으로 bind()한 buffer object를 상태로서 들고 있는 거죠. 한 예를 들면, 화면에 삼각형들을 그리고 싶다, 그런데 어떤 정점 배열을 참조하지? 할 때 마지막으로 bind()된 정점 배열 버퍼의 내용을 화면에 뿌려준다, 이런 식으로 동작한다고 할 수 있어요.

위 내용에 기반해서, buffer object들은 각 종류별로 서로 다른 클래스로 묶어줄 겁니다. 각각은 자신의 ID, 생성, 제거, bind 메서드를 공통으로 들고 있을 거고요. Buffer object의 ID가 만들어지는 과정 자체가 API의 생성 콜을 요구하므로 생성 메서드는 static으로 해서 그 클래스 자신을 리턴하는 형식을 쓰는 것이 좋습니다 Code 1 참조.

```

public class Object
{
    Object(int arg1, int arg2) { /* initialization */ }
    public static Object CreateObject(int arg)
    {
        /* processing */
        return new Object(arg1, arg2);
    }
};

```

**CODE 1.** 이런 형태의 CreateObject같은 거

## B. Vertex Buffer Object(VBO)

VBO는 정점의 속성들을 들고 있는 버퍼입니다. VBO 하나당 속성 하나씩 넣어서 나중에 뚫느냐 VBO 하나에 모든 속성 데이터를 다 집어넣느냐는 상당히 취향이 갈리는 문제인 거 같긴 한데, 마지막으로 렌더링 엔진을 만들었을 때는 VBO 하나에 모든 속성 데이터를 넣는 방식을 취했어요. 그 땐 그렇게 해야 물체 하나당 정점 속성 배열 하나가 나와서 VAO(뒤에 다룹니다)의 생성 메서드가 깔끔해진다는 이유였던 것 같은데, 그 디자인 선택이 나중에 **III. 정점 속성**에서 다룰 정점 속성 관련 난제를 만들었습니다. 나중에 한번 더 만든다면 전자를 시도해보고 싶긴 해요.

VBO 생성 메서드를 부를 때 정점 속성들 전부<sup>2</sup>와 이 버퍼를 어떤 용도로 사용할지 `bufferUsageHint`를 인자로 받도록 하면 buffer object를 만드는 데에 충분한 정보를 얻을 수 있습니다. 계속 들고 있어야 하는 건 API에서 부여받은 ID. 파티클 시스템이라도 만든다면 혹시 나중에 정점 정보를 업데이트해야 할지도 모르니 정점 개수와 사용 용도를 들고 있는 것도 좋습니다<sup>3</sup>.

## C. Element Buffer Object(EBO)

EBO는 정점들을 어떤 순서로 그릴지에 대한 정보를 들고 있는 버퍼입니다. 정점 배열에서의 `index`들을 순서대로 갖고 있는 배열만 받으면 충분합니다. 이 친구는 ID뿐만 아니라, 나중에 렌더할 때 필요하니 생성할 때 받은 `index` 배열의 크기도 갖고 있어야 합니다.

---

<sup>2</sup> 자세한 정보는 **III. 정점 속성** 참조

<sup>3</sup> Static인데 업데이트하고 그러면 안 되잖아요.

## D. Vertex Array Object(VAO)

VAO는 복수의 VBO와 하나의 EBO를 서로 묶어주는 역할을 하는 오브젝트입니다—만은 우린 많은 정점 속성을 하나의 VBO에 넣고 있기도 했으니까 각각 하나씩만 들고 있으면 충분합니다. 물체를 그리는 데에 필요한 모든 속성(정점들의 속성과 그릴 때의 순서)을 모두 받아 VBO와 EBO의 `instance`를 생성해 들고 있도록 합시다. VAO 자신의 ID도 물론 들고 있어야겠습니다.

이제 `drawable` 3D 물체에 자신의 VAO `instance`를 갖고 있게 하고 그걸 `bind()`한 후 `draw()`하는 것으로 화면에 그 물체를 그리게 할 수 있습니다. 음. 비교적 간단해졌네요(정말?).

## III. 정점 속성

OpenGL 감싸줄 때 가장 고민 많이 했던 부분이 여깁니다. `glBufferData`로 정점과 관련된 데이터를 GPU에 업로드해줄 때 데이터 배열 포인터와 사이즈를 던져주면 그 메모리 레이아웃 그대로 올려버리거든요…(멤버 함수같은 게 있다면 그대로 공간이 낭비됩니다). 데이터를 올린 후에 그 데이터에서 어느 부분이 어떤 속성에 대한 정보를 갖고 있는지 역시 `glVertexAttribPointer`로 알려주어야 하는데(이 역할을 하는 API 콜의 연쇄를 `setup`이라 부르겠습니다), 정점 속성 집합<sup>4</sup>이 서로 다르면 이것도 각각 달라야 하고요.

그래서 렌더링 엔진에서 사용할 정점 속성 집합들의 종류를 제한하고, 이들을 각각 개별의 `struct`로 명시합니다(예를 들자면 정점의 위치와 `normal vector`, 색깔 데이터를 들고 있는 `VertexPositionNormalColor` 따위). 만약 C#처럼 `reflection`이 지원되는 언어라면 해당 타입에 맞는 `setup`을 불러주는 `helper`를 구현할 수 있겠고, 아니라면 각 정점 속성 집합 `type`마다 데이터 배열과 그에 해당하는 `setup`을 갖는 `class`를 또 만들어 VBO `instance`를 생성할 때 그걸 넘겨줄 수 있겠죠 [Code 2](#) 참조. 그걸 받아서 어떻게 할지는 VBO 측 생성 메서드에서 알아서 처리하도록 하고.

여러 종류의 정점 속성을 사용하다 보면 쉐이더와의 관계 역시 골치아플 수 있는데, 워낙 다양한 쉐이더 코드를 다양한 정점 속성 집합에 대해 공용으로 사용할 일이 많다 보니 그냥 렌더링 엔진에서 사용할 모든 속성들에 대해 일련번호를 붙이고 그걸 다 같이 쓰기로 합의하는 것도 괜찮은 방법인 것 같습니다. `World space`에서의 위치는 0번, `normal vector`는

<sup>4</sup> 어떤 정점이 가지는 속성의 집합. 표현하고 싶은 것에 따라 서로 다른 속성들을 들고 있을 수 있으니까요. 어떤 정점은 `world space`에서의 위치와 색깔, `normal vector`가 필요하지만 어떤 정점은 `screen space`에서의 위치와 색깔만 필요하다던가.

```

public class VTypeWrapper : VTypeWrapperBase
{
    VType *data;
    // overrides virtual function of VTypeWrapperBase
    public size_t stride()
    {
        return sizeof(VType);
    }
    // overrides virtual function of VTypeWrapperBase
    public void setup()
    {
        /* setup */
    }
};

```

**CODE 2.** 대충 이런 느낌의 wrapper

1번, 이런 식으로요. 쉐이더 코드에서도 `layout (location = x)` qualifier를 사용해서 컴파일/링크 전에 속성의 위치를 명시해줄 수 있거든요.

다시 생각하는 거지만 VBO 하나당 하나의 속성만 들어가게 했다면 뭔가 좀 더 깔끔하게 만들 수 있지 않았을까 하는 생각이 자꾸 드네요. 그럼 '정점 속성 집합'같은 개념도 필요 없을 거고, `offset`은 0이요 `stride`는 그 속성의 `size`이니 `setup`과 관련된 추가적인 레이어도 사라질 테니까요.

## IV. 쉐이더 프로그램

정점 데이터를 받아서 화면에 어떻게 뿌려질지를 정하는 건 쉐이더죠. 쉐이더 코드들을 컴파일/링크해서 얻을 수 있는, 프로그램 안의 프로그램을 **쉐이더 프로그램**이라고 합니다. 따라서 모든 쉐이더 프로그램을 만드는 데에 사용되는, 쉐이더 코드 컴파일링과 링킹을 하는 메서드를 갖는 베이스 클래스를 만들어둡시다. 이 베이스 클래스에는 또한 `glUseProgram`을 불러줄 `use()`메서드와 쉐이더 프로그램으로부터 `uniform` 위치를 얻어올 추상 메서드(쉐이더마다 다를 테니까요), `fragment data`를 `bind`할 가상 메서드(동시에 여러 개의 렌더 타겟을 쓸 수도 있으니까요)도 포함됩니다.

아제 서로 다른 목적을 가진 쉐이더 프로그램의 클래스는 이 베이스 클래스를 상속받아 만들어집니다. 자신이 위치를 얻어오고 싶은 `uniform`을 명시해 주고, 각 `uniform`에 들어갈 값을 넘겨주는 메서드가 추가됩니다. `Uniform`에 값을 넘겨줄 때 참조해야 하니까 각 `uniform`의 위치는 들고 있어야겠죠.

이제 렌더러에서는 어떤 쉐이더 프로그램을 사용해 렌더할 때 이 클래스의 `instance`를 `use()`하고 `uniform`에 적절한 값을 넘긴 후에 렌더를 하면 됩니다.

## V. Framebuffer와 Texture

Texture와 framebuffer는 마치 종이와 틀 같다고 할 수 있습니다. 만약 당신이 텍스처에 렌더를 하고 싶다면 더욱요.

**Texture**를 감싸는 클래스는 앞에서 다루었던 buffer object와 크게 다르지 않습니다. 차이가 있다면 `texture`에 대한 `bind()`인 `glBindTexture`는 같은 `texture`에 대해서도 `bind target`이 달라질 수 있다는 점, 그리고 생성시의 가능한 동작으로 이미 있는 2D 비트맵 이미지를 가져오는 것 외에 비어 있는 `texture`로 생성하는 것이 있다는 점 등입니다. 각자 취향과 상황에 맞게 `bind()`와 생성 메서드를 디자인해 주세요.

**Framebuffer**를 사용하고 싶다면 `texture`는 자신이 나타내고자 하는 것이 `color`인지 `depth`인지를 들고 있는 것이 좋습니다. Framebuffer(이 친구도 buffer object이므로 앞서 거론한 형태에서 크게 벗어나지 않습니다)의 생성 메서드가 그 framebuffer에 붙이고자 하는 `texture`들의 목록을 받을 텐데, 그 때 `color attachment`에 붙여야 할지 `depth attachment`에 붙여야 할지를 알아야 하기 때문이죠.

0번 framebuffer는 미리 예약되어 있는 system framebuffer(네, 모니터 화면이요)이기 때문에, 0번 framebuffer를 `bind`하는 `Unbind()` 메서드를 구현해 두면 추후에 편리합니다.

## VI. Drawable

지금까지 열심히 화면에 그리기 위한 도구를 래핑했습니다. 이제 화면에 그려질 수 있는 모든 걸 뭉뚱그린 무언가가 필요하죠! `Drawable`은 VAO instance와 그걸 `bind()`한 후 `glDrawElements()`하는 `draw()`메서드를 갖고 있는 친구입니다. 기호에 따라 이걸 상속받아 static mesh를 그리는 오브젝트를 만들거나, dynamic particle system을 만들거나 할 수 있어요. 정말 별거 없네요.

하지만 원래 그런 거 같아요. 어떻게 하면 잘 감쌀까에 대한 얘기였으니까, 로우레벨에서 점점 위로 올라가는 순서로 쓰인 이런 형식의 글에선 뒤로 갈수록 쓸 게 없어지는 게 좋은 거 아닐까요? 추상화가 잘 됐단 증거겠죠. 만세!

## VII. 렌더러

렌더링 엔진의 마지막 단계입니다. 렌더러는 원하는 렌더 결과를 얻기 위해 다수의 쉐이더 프로그램과 framebuffer를 적절하게 사용합니다. 또한 그려야 할 drawable도 프레임마다 갱신받아야 하죠. 카메라처럼 렌더링 과정 전체를 통틀어 일정하게 함께하는 추가정보 역시 들고 있어도 좋습니다.

먼저 렌더러를 초기화할 때 이 렌더러가 사용할 쉐이더 프로그램과 framebuffer를 초기화해서 들고 있게 합니다. 한 프레임에 여러 번 렌더 콜을 부르지 않게 하기 위해 그릴 drawable을 enqueue하는 메서드를 작성하고, 한 프레임 안에 그려져야 할 모든 drawable이 모였을 때 부를 render() 메서드를 작성합니다.

이 render() 메서드 안에서는 지금까지 래핑해온 buffer object나 쉐이더 프로그램, framebuffer 등을 자유롭게 활용하여 원하는 렌더 결과가 나올 때까지 iterate해요. 래핑하는 데에 좀 오래 걸리긴 했지만, 이 때 와서 iterate할 때 느끼는 능률 향상은 상당히 뚜렷합니다.

```
void render()
{
    colorFBO.Bind();
    {
        /* FBO init */
        colorShader.Use();
        foreach (var mesh in meshList)
        {
            colorShader.SetUniform
            (
                /* Uniform Setting */
            );
            mesh.Draw();
        }
    }
    colorFBO.Unbind();
    ...
}
```

**CODE 3.** 이쁘게 감싸면 기분이 조크든요

이제 메인 인터랙티브 어플리케이션에선 이 렌더러와 그려야 할 drawable들을 초기화하고, 매 프레임마다 렌더러에 그려질 drawable들을 전송한 후 render()하면 렌더가 진행됩니다.

## VIII. 클래스 다이어그램

지금까지의 추상화를 그림으로 그려보면 대략 다음 Figure 1과 같습니다.

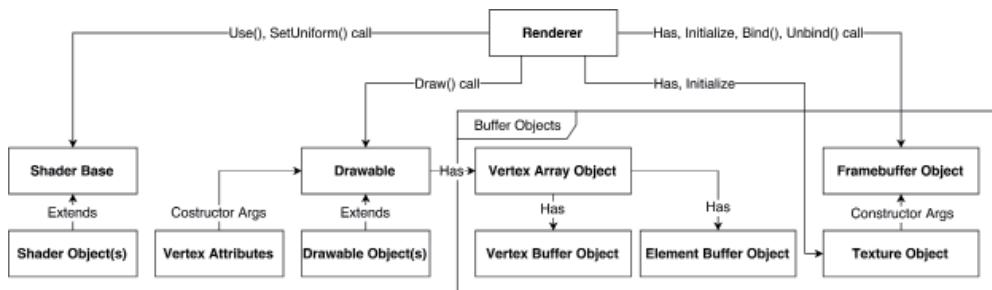


FIGURE 1. 대충 이런 느낌입니다

## 참고사항

<https://bitbucket.org/haje/mmo>에서 관련 코드를 열람하실 수 있습니다.

/Client/Rendering안을 적절히 뒤져보시면 적절히 나옵니다. 읽다가 대체 무슨 얘기를 하는 건가 싶을 때 들어가서 비교분석하면서 읽으시면 도움이 될지도 모릅니다.

## Acknowledgements

게임 제작 및 렌더링 엔진 개발 경험을 할 기회를 제공해 준 KAIST 게임제작동아리 HAJE와 MMO 프로젝트 참여진들께 감사드립니다. 특히 프로젝트를 이끌어 준 디렉터 도우진 님과 함께 렌더링 관련 프로그래밍을 한 그래픽스 프로그래머 김의태 님, 래핑과 구조화에 큰 도움을 준 리드 프로그래머 이준희 님께 큰 감사를 드립니다. ssc 1st

귀평을 겠다  
신운널리  
대여셔 드습니다

Functional 몸에 좋아요.  
해치지 않아요.





Microsoft C# (2000 ~ )

Console.WriteLine("Hello,  
world!");

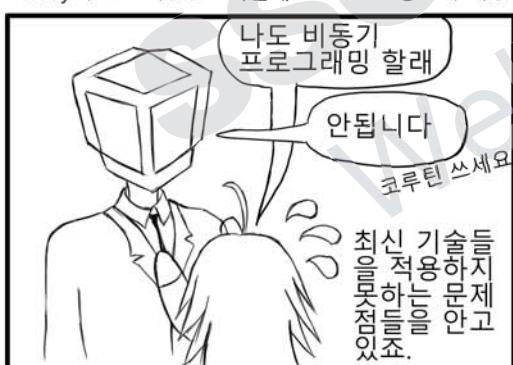


대신귀여운평셔널을드리겠습니다

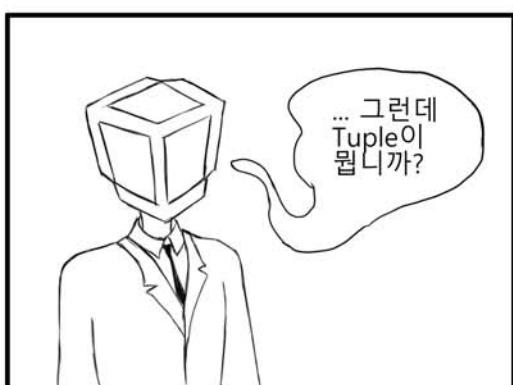


\* Unity의 모노버전은 2버전대로 .Net 3.5정도에 해당

\* Abstract Equality Comparison Algorithm 참조



\* F#은 다른 .Net 언어와의 작업을 위해 한정적으로 null 허용



\* Tuple은 .Net Framework 4.0에서 추가.



\* C#에서는 null과 숫자나 boolean값과는 비교불가 (컴파일 에러)



```
let rec factorial n =
  match n with
  | 0 -> 1
  | _ -> n * factorial (n-1)
```

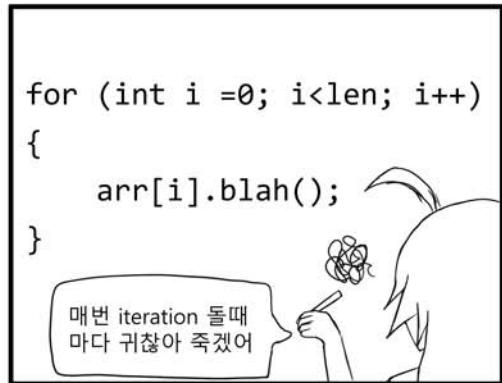
... Hello World 도  
해보라구요?

printfn "Hello, world!"

F# 이에요. 2006년에 나온 함수형 언어로  
프로그래밍 원리 자체를 연구하는 언어에요.



F# (2005~)



\* 실제로는 arr.foreach(member => member.blah()) 같은 식으로 씁니다.



\* 아동범죄 신고는 언제나 112

# 주토피아는 실재하는가

인류사를 통해 바라본 수인 종족의 다양성에 대한 담론

Ria Fox

## I. 서론: 태초에 수인이 있었다

많은 창작물에서 수인을 등장시키면서 이제 수인은 우리에게 낯설지 않은 존재가 되었다. <던전 앤 파이터>의 겹쟁이 사막여우 수인이라던가, <강철의 연금술사>의 ‘하인켈’ 등 게임과 애니메이션 등에 수인은 조연으로, 때때로는 주연으로 등장한다. ‘동물의 모습을 가지고 있으면서 인간의 지능을 가지고 있다면 어떤 모습과 행동을 보여주게 될 것인가’라는 아주 예전부터 있었던 인류의 궁금증과 상상에 대한 산물로서 나타난 수인은 모습 그대로 ‘동물과 인간의 특징을 모두 가진’ 종족으로 인식되고는 한다. 실제로 여러 창작물에서 그들은 사회를 이루고, 문명을 이룩하였으며 현대의 인간과 다르지 않은 모습을 많이 보여준다. 지능 또한 인간과 같을 것이고, 인류와 비슷한 문명의 역사를 가졌을 것이라고 생각된다. 하지만 인간 또한 태

초에는 여러 종이 있었다. 그리고 지금은 단 한 종만 남게 되었다. 그렇다면 수인종이 실재한다는 가정 하에 그들은 어떤 길을 걷게 될 것인가?

## II. 동물과 인간, 그리고 수인

우선 수인이 동물이나 인간과는 다른 특성을 가지고 있음을 짚고 넘어가야 한다. 수인은 외모, 인간과 동물 사이를 넘나드는 생활 패턴, 행동 때문에 흔히 동물과 인간의 중간 정도의 생명체로 많이 알려져 있다. 많은 창작물에서도 인간과 동물의 유전자 조작이나, 인간의 변화에 의한 생물, 또는 지능이 있는 동물 정도로 표현된다. 물론 수인은 동물의 신체적 특징과, 인간의 골격과 지능을 가지고 있기는 하지만 오리너구리의 예를 보면 신체적인 특징을 제외한 행동 패턴이나 특성이 완벽히 다른 별개의 생물일 수 있다는 가능성은 시사한

다.

오리너구리는 오리와 너구리의 중간 정도에 있는 생물인 것 같지만, 실제로는 바늘두더지Genus *Tachyglossus*와 태반류, 그리고 유대목과 같은 공통 조상을 가지고 있다 [1]. 또한 포유강 식육목에 속하는 너구리와 조강 기러기목에 속하는 오리와는 달리 포유강 단공목에 속하는 오리너구리는 너구리와 오리의 생활 패턴 또는 특성을 전혀 닮지 않았다. 형태를 보더라도 주둥이가 오리와 비슷하게 생겨 “오리너구리”라는 이름을 얻게 되었지만 몸은 수달과 비슷하며 꼬리는 비벼와 닮았으며, 강이나 연못 등지에서 물고기를 잡아먹는 오리나 잡식성인 너구리와 달리 늪지에 살며 늪지 바닥을 부리로 파서 민물 새우나 곤충 유충, 민물 가재 등을 먹고 사는 육식성이다[2]. 이와 같이 수인 또한 그 모티브가 되는 동물과 인간과는 완벽히 다른 생태를 가질 수 있으며, 그렇기에 단순히 동물과 인간 사이의 종족으로 보는 것은 큰 오류를 범할 수 있다.

그렇기에 수인은 인간과는 완벽하게 같은 역사를 걸어올 수 없다. 몸의 구조와 특성에 따라서 같은 인간이라도 다른 생활 패턴을 가질 수 밖에 없으며, 그로 인해서 발달하는 문화는 꽤 많은 차이를 보인다. 예를 들면, 동북아시아를 제외하면 대부분의 사람들은 몸에 악취가 나는 것을 당연하게 여기는데, 이는 사람마다 가지고 있는 유전자의 차이 때문이다[3]. 액취증을 유발하는 대립 유전자 G와 아포크립땀샘의 분비가 적은 대립 유전자 A의 차이로 데오드란트 문화라는 동북아시아에서는 쉽게 볼 수 없

는 문화를 만들었듯이 수인과 인간은 같은 듯하면서 다른 문화를 쌓아 나갔을 것이라고 예상할 수 있다.

그렇기 때문에 우리는 수인을 동물과 인간의 생활 패턴을 조금씩 가지고 있으면서, 실제로는 그들만의 새로운 특성을 가지고 있는 종족으로 생각을 해야 한다. 인간의 특징과 동물의 특징, 그리고 수인만의 특징을 얼마나 가지고 있을 것인가를 생각할 수 있으며 문명을 이룩한 후에 동물의 특징을 버릴 것인가, 또는 문명을 포기하고 인간의 특징을 버릴 것인가도 고려할 만 하다.

고려해야 할 것 중 다른 하나는 수인 종족 중 개와 현대의 소, 그리고 돼지와 같은 ‘가축’ 종은 존재하지 않는다는 것을 염두에 두어야 한다는 것이다. 개는 늑대를, 소는 들소를, 돼지는 야생 멧돼지 중 인간에게 친근한 종만 길들여 인간이 가축화 한 종들로 자연적으로는 분화되지 않는 종들이기 때문이다. 현재 가축화된 동물들은 대부분 야생에서 살아남기 힘든 신체적 특징을 가지고 있는데, 그 예 중 하나가 바로 도베르만이다. 도베르만 핀셔라는 품종을 만들기 위해 인간은 몇 번이나 근친교배를 시켰고, 그로 인해서 dilated cardiomyopathy, cervical vertebral instability(CVI), von Willebrand's disease, prostatic disease와 같은 병을 얻을 확률이 심각하게 높아지며[4], 다른 종에 비해서도 갈비뼈가 지나치게 커 쉽게 다치기 때문에 도베르만이라는 종은 야생에서 먹이 경쟁에 밀릴 수 밖에 없다. 다양한 종류의 고양이 또한 마찬가지다. 하지만 고양이의 경우 오늘 날 집고양이의 조상으로 믿어지는 근동 야

생고양이 *Felis silvestris lybica* 종은 존재할 것이라고 추측된다.

### III. 인지혁명: 반목하는 종족들

늑대 수인, 고양이 수인, 토끼 수인…… 수인의 종류는 우리가 생각할 수 있는 동물 수만큼이나 많다. 어쩌면 두 개 이상의 동물이 섞여 있는 모습일 수도 있고, 어쩌면 우리가 상상도 하지 못한 새로운 동물의 수인일 수 있다. 영화 <주토피아>와 같이 많은 종류의 수인이 서로 조화롭게(어쩌면 차별이 있을 수도 있겠지만) 생활하는 세계를 꿈 꿀 수도 있을 것이다. 하지만 근본적인 질문을 던져 보자. 이렇게 많은 종의 수인들이 한 세계에 조화롭게 공존할 수 있을까? 인류학의 관점에서 보자면 답은 ‘가능성은 있지만, 다양한 종족이 존재하기 힘들다’이다.

현재 지구상에 남은 유일한 인간 종은 ‘호모 사피엔스’이다. 지난 1만 년간 우리 종은 지구상의 유일한 인간 종이었기 때문에, 우리는 스스로를 유일한 인류라고 생각하는 데에 익숙해 있다. 하지만 ‘인간’이란 말의 진정한 의미는 ‘호모 속에 속하는 동물’이고, 호모 속에는 사피엔스 외에도 여타의 종이 많이 존재했다[5]. 인류는 약 250만년 전 동부 아프리카의 오스트랄로피테쿠스에서 진화했으며 그들은 고향을 떠나 여행을 시작해 북아프리카, 유럽, 아시아의 넓은 지역에 정착하여 서로 다른 여러 종들이 생겨났다. ‘호모 네안데르탈렌시스’, ‘호모 에렉투스’, ‘호모 솔로엔시스’ 등의 적어도 6종의 다른 호모 속의 인간이 있었다. 하지만 호모 사피엔스가 동아프리카를 벗어나

유라시아 대륙으로 건너가기 시작하면서 다른 인간 종은 점점 사라져갔다. 이들에게 무슨 일이 있었는지 설명하는 이론으로는 ‘교체이론’과 ‘교배이론’이 있는데, 교체이론은 서로 다른 종의 인간들이 서로 화합하지 못하고 반감을 보였으며, 심지어 인종 학살이 일어났다는 이야기이다. 이 이론에 따르면 사피엔스와 다른 인간 종들은 해부학적으로 달랐으며 짹짓기 습관이나 체취 까지도 차이가 났을 가능성이 매우 커서 서로에게 성적인 관심을 거의 느끼지 못했다. ‘교배이론’은 교체이론과 대립되는 견해로, 그들이 서로 끌려 성관계를 하고 뒤섞였다는 설이다. 최근 몇십 년은 교체이론이 이분야의 상식이었으나, 계놈 프로젝트로 인해 실제로는 네안데르탈인의 유전자와 데니소바인의 유전자가 각각 중동, 유럽에 거주하는 사람들과 멜라네시아인, 호주 원주민에게서 발견되었기 때문이다.

수인의 경우 서로 다른 종의 수인은 같은 종이 아니다. 그들은 서로 섞일 수 없고, 그렇기 때문에 교배이론을 적용할 수 없다. 하지만 교배이론이 교체이론을 대신하였다고 해서 사피엔스라는 종이 다른 종과 평화롭게 어울린 것은 아니었다. 다른 종이 멸종한 이유에 대한 가능성으로, 사피엔스의 기술과 사회적 기능이 우수한 덕분에 사냥과 채취에 더 능숙했고, 그렇기 때문에 사피엔스 집단에 합류한 한두명의 예외를 제외하고는 전부 멸종했기 때문에 사피엔스가 유일한 인간 종으로 남았다는 가능성 또한 있지만, 자원을 둘러싼 경쟁이 폭력과 대량학살을 유발했다는 가능성 또한 존재한다. 현대의 경우를 보아도 사피엔스 집단은 피부

색이나 언어, 종교의 작은 차이만으로도 곧 잘 다른 집단을 몰살하는 것처럼, 원시의 사피엔스라고 해서 자신들과 전혀 다른 인간 종에게 이보다 더 관용적이었다고 생각되진 않는다.

사피엔스가 다른 인간 종들과 차별화되었던 점은 바로 인지혁명이다. 인지혁명으로 인해 규모가 더 크고 응집력이 더 강한 집단을 만들 수 있었고, 또한 대단히 많은 숫자의 낯선 사람들끼리 협력이 가능했다. 그렇기 때문에 네안데르탈인과 같이 사피엔스보다 더욱 뇌의 용량이 크고 신체가 강한 종족이라고 하더라도 수로 밀어붙이며 빠른 정보 전달이 가능했던 사피엔스에게는 역부족이었다. 하지만 수인종들은 모두 사피엔스와 같은 지능을 가진다고 앞에서 가정했었다. 이 말은 각각의 수인종에게 있어서 집단과 전술의 능력 차는 없다고 봐도 무방하다는 것이다. 동물들과는 달리 수인에게는 이족보행과 도구라는 발톱과 이빨을 대신할 무기가 존재하고, 이 때문에 육식성 수인과 초식성 수인의 절대적인 전투 능력의 차이는 발생하지 않는다. 하지만 이 가정은 모든 수인종이 사피엔스만큼 큰 집단을 가졌을 것이라는 가정에서 나온 것이다. 대부분의 수인종들이 문명 발달 이전에는 베이스가 되는 동물의 특성을 어느 정도 가지고 있다고 가정한다면, 대부분의 수인종은 그들의 베이스가 되는 동물의 식성을 따라갈 것이다. 그렇다면 먹이 피라미드에 의해 육식성 수인들은 초식성 수인보다 더 적은 숫자가 존재할 수 밖에 없다. 또한 적은 무리가 넓은 고정된 영역을 가지는 육식성 수인과 달리 초식성 수인은 고정된 영역

을 가진 종 또한 존재하지만, 바이슨과 같이 많은 무리가 유동적인 유랑 생활을 하는 종족 또한 존재한다[6]. 즉, 절대적으로 육식성 수인은 수가 부족할 수 밖에 없다는 것이다.

그렇다면 여우와 같은 잡식성 수인의 경우 적은 수의 육식성 수인의 집단 형태를 가질 것인가, 아니면 초식성 수인의 집단 형태를 따를 것인가? 원래의 사피엔스와 같은 식성을 가진다면 그 집단은 사피엔스의 집단과 가장 비슷할 것이다. 육식성 수인이 수렵을 하고, 초식성 수인이 채집을 한다면 잡식성 수인은 수렵과 채집을 모두 할 수 있다는 말이다. 이 경우 초식성 수인이나 육식성 수인에 비해서 좀 더 유동적인 식단을 가질 수 있고, 그로 인해서 잡식성 수인의 집단이 가장 클 것이라는 이야기가 된다.

집단의 크기만으로는 어떤 종의 멸종이 확정되는 것이 아니다. 그들이 가진 도구와 식량 등의 전쟁 지속 능력 또한 중요하기 때문이다. 하지만 번식 능력이 경쟁자들에 비해 지나치게 부족하거나 종족 내의 번식이 엄격하게 제한된다면 네안데르탈인이 멸종한 것과 같이 도망쳐 다른 곳에서 새로운 터를 잡기 전에 전부 멸종하게 될 것이다. 이런 이유로 상대적으로 번식력이 밀리는 종족은 농업 혁명 이전에 멸종될 것이다.

## IV. 농업 혁명과 국가의 탄생

사피엔스, 네안데르탈인의 전쟁과 수인종끼리의 전쟁은 많은 차이가 있다. 사피엔스는 언어가 있었고, 결집력이 있었으며, 실

제로 존재하지 않는 것들에 대한 정보를 전달하는 능력이 있었다. 그렇기 때문에 네안데르탈인은 굉장히 빠르게 사피엔스에게 인류의 왕 자리를 넘겨줄 수 밖에 없었다. 하지만 모두가 사피엔스의 능력을 가지고 있는 수인종들이라면 이야기가 달라진다. 이들의 전쟁은 절대 사피엔스와 네안데르탈인의 경쟁만큼이나 빠르게 끝나지 않을 것이기 때문이다. 서로의 병력 차가 있더라도 그들에겐 언어가 있다. 서로 다른 수인종끼리 결집할 수 있으며, 이 차이는 전쟁에 있어서 많은 것을 바꾸게 된다. 그렇기 때문에 농업혁명 전까지는 인류가 걸어왔던 길과는 달리 전국시대의 중국과 같이 수많은 작은 동맹들이 존재하는 혼돈의 시대였을 것이다.

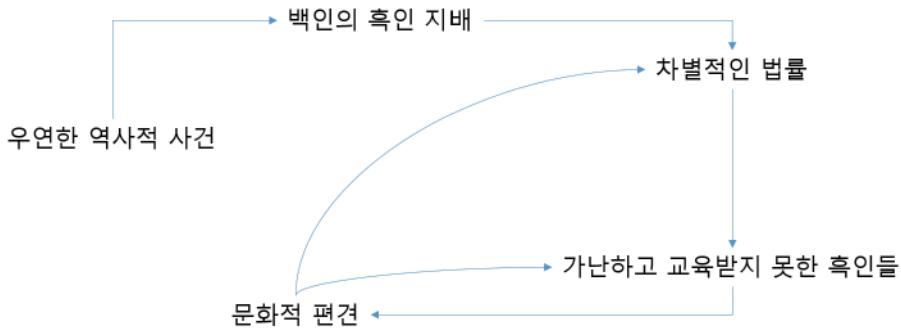
하지만 농업혁명이 일어나는 순간 그 균형은 깨지게 된다. 농업 혁명으로 인해 잡식성 수인과 초식성 수인이 폭발적인 인구 증가를 할 수 있는데 반해 육식성 수인은 농업혁명의 영향을 거의 받을 수 없기 때문이다. 농업 혁명으로 인해 동물의 가축화가 일어났지만, 가축을 먹이기 위해서 농사를 짓는다는 선택은 하지 않았을 것이다. 그 이유로는 당시의 농업 효율로는 생산한 곡식이 집단의 모든 사람들을 먹여 살리기 위한 최소한의 가축을 먹여 살릴 수 없었기 때문이다. 그렇기 때문에 유량 생활을 선택했을 텐데, 유량 생활은 농업 혁명 이전의 수렵채집인들의 생활과 비슷한 수의 집단만을 이룰 수 있었을 것이다. 가축의 수가 늘어나면 늘어날수록 이를 관리할 방법이 필요하고, 또한 가축 한 마리당 곡식에 비해서 사람을 먹여 살릴 수 있는 양은 제한

되어있기 때문이었다. 그렇기에 육식성 수인들은 기존과 비슷한 인구수를 가지게 되었을 것이다.

하지만 모든 육식성 수인은 멸종할 것인가라는 질문에 대한 답은 ‘그렇다’가 아니다. 원래 한정된 자원을 가지고 경쟁하던 수인들이 이제는 한정된 토지를 가지고 경쟁하도록 경쟁의 양상이 바뀌기 때문이다. 원래는 고정된 넓은 지역을 영역으로 하던 육식성 수인들이 이제는 넓은 땅을 돌아다니는 유목 생활을 할 수밖에 없을 것이고, 고정된 지역에서 살던 초식성 수인과 유량 생활을 하던 초식성 수인들, 그리고 잡식성 수인들은 농사를 짓기 시작할 것이다. 몽골의 유목 민족들이 주변의 정착 국가들에 의해서 멸종하지 않은 것과 마찬가지로 그들은 그들 나름의 정착민들과 다른 기술과 생활을 이어나갈 수 있기 때문이다.

이제 ‘모든 수인종이 비슷한 힘의 크기를 가지고 있고 전국시대와 같은 국가가 생성되었으니, 이제 모든 수인종은 그들만의 문화를 가지고 멸종되지 않고 현대까지 남을 것이다’라고 생각할 수도 있을 것이다. 하지만 그것은 오산이다.

농업혁명 이후 수천 년에 이르는 인간의 역사를 이해하려는 시도는 단 하나의 질문으로 귀결된다: 인류는 어떻게 자신들을 대규모 협력망으로 엮었는가? 그런 망을 지탱할 생물학적 본능이 결핍된 상태에서 말이다. 결론적으로는, 불평등이 생겨났고 사람들을 서열로 구분된 가상의 집단으로 나눔으로서 가능했다[7]. 이런 계급 구조는 우연한 역사적 사건으로부터 시작되어 악순환을 만들게 된다Figure 1 참조. 수인종의



**FIGURE 1.** 미국의 사회적 악순환: 우연한 역사적 사건은 견고한 사회구조로 변했다.

세계에서도 이런 현상은 똑같이 일어날 것이다. 거기다가 수인종의 세계는 인지혁명 이후의 결과로서 서로 다른 종이 협력해서 만든 나라가 아닌, 살아남기 위해서 단일 종 또는 2, 3개의 종족만으로 이루어진 나라일 것이다. 이 두 가지가 합쳐진다면 ‘한 종족의 모든 개체의 노예화’라는 치명적인 결과를 낳게 된다. 인류의 대부분의 사회정치적 차별에는 논리적, 생물학적 근거가 없으며, 우연한 사건이 신화의 뒷받침을 받아 영속화한 것에 불과하다. 하지만 만일 실제로 계급의 구분이 생물학적 실체에 근거를 두었다면 어떨까? 인간 사회에서는 계놈 프로젝트 이후에 ‘모든 호모 사피엔스의 각기 다른 집단이 지니는 생물학적 차이는 사실상 무시할 만한 수준’이라는 것이 밝혀졌지만, 수인종에게는 그렇지 않다. 한 번 노예가 된 종은 과학의 뒷받침을 받아 더욱 더 비참한 생활을 하게 될 것이다.

하지만 이에 대해서 다른 해석을 내 놓을 수도 있다. 인류학에서는 1960년 이후 종족 집단과, 종족 집단을 구성하는 특질을 가리키는 종족성ethnicity이라는 용어가 부족주의tribalism를 대체하여 등장하였다. 종족성은 하나의 완벽한 정의로 규정하기

는 힘들지만, 다양한 정의에 내포되어 있는 종족성의 여러 측면들을 살펴보는 것에 의미가 있다. 종족은 인종의 범주와도 일부 중첩되고, 민족의 관념과도 중첩되는 용어지만 여기서는 수인종의 신체적 특징이 실제로 다르므로 종족의 개념을 인종의 개념으로 볼 수 있다[8]. 같은 종족성을 가지고 있다면 같은 종족이 아니더라도 서로를 이해하고 쉽게 어울릴 수 있다. 한국계 미국인 인류학자인 샌드라 리는 재일동포 가운데 노년 세대를 연구하면서 이들이 신체적 특징상으로 별 차이가 없고, 문화적으로도 유사한 일본 사회에서 자신들의 종족 정체성을 어떻게 유지하는가를 연구하였는데, 같은 한국계 후손이지만 언어나 문화적으로 공통점이 별로 없는 자신을 재일동포 노인들이 어떻게 받아들일지 몰라 걱정하였지만 자신이 매운 김치를 먹는 모습을 보고서야 그들이 자신을 한국인으로 인정해 주었다고 진술한 바 있다[9]. 이와 같이 같은 종족성을 가지고 있는 종족이라면 같은 국가를 이룩하는 것은 물론 서로 차별없이 지낼 수 있는 발판이 마련되는 것이다. 하지만 종족성을 생각하더라도 수인종의 식성은 종족성에 굉장히 큰 영향을 끼칠 것이

고, 이는 육식성 수인과 잡식성 수인, 초식성 수인의 차별을 가속화 시킬 수 있기 때문에 종족성의 관점에서 보아도 차별은 사라질 수 없다.

## V. 결론 : 디스토피아적 세계

극장에서 보던 주토피아는 육식—초식 동물들의 미묘한 차별이 있긴 했어도 결국 해결되는 해피엔딩으로 끝났다. 하지만 실제의 수인 세계는 극장에서 보던 주토피아가 아닌 그 예전 스토리와 같은 디스토피아일 것이다.

육식동물이 차별을 받던지, 초식동물이 차별을 받던지, 아니면 어떤 종족이 차별을 받던지 어쨌든 어떤 종족은 노예와 같이 살아갈 것이다. 심지어는 과학의 힘을 빌리더라도 그들의 차별은 심화되기만 할 것이다. 현재의 인류가 차별 없는 사회를 꿈꾸게 된 것은 순전히 인간 사이의 생물학적 차이가 없기 때문이다. 여기에 실제로 생물학적 차이가 있는 수인 세계라면 독일의 우생학이 정설로 받아들여질지 모른다. SSCC 1st

## References

- [1] G. Lecointre and H. L. Guyader, *The Tree of Life: A Phylogenetic Classification* (Harvard University Press, Cambridge, 2006).
- [2] P. Bethge, Ph.D. thesis, University of Tasmania, 2002.
- [3] R. Hart, *Nexus* 1, 1 (1980).
- [4] D. R. Krawiec and D. Heflin, *J Am Vet Med Assoc* 200, 1119–22 (1992).
- [5] Y. N. Harari, 사피엔스: 유인원에서 사이보그까지, 조현욱 역 (김영사, 파주, 2015), p. 22.
- [6] V. G. Heptner and N. P. Naumov, *Mammals of the Soviet Union* (Smithsonian Institution Libraries and National Science Foundation, Washington, D.C., 1998), Vol. 2, Part 1a, p. 222.
- [7] Y. N. Harari, 사피엔스: 유인원에서 사이보그까지, 조현욱 역 (김영사, 파주, 2015), p. 196.
- [8] 한국문화인류학회, 처음 만나는 문화인류학 (일조각, 서울, 2003), p. 126-129.
- [9] 한국문화인류학회, 처음 만나는 문화인류학 (일조각, 서울, 2003), p. 129-130.

## ○○위키가 알려주지 않는 수학 이야기

小栗戸栗 (Ogri Togri)

팬티를 입는 방법은 몇 가지나 될까? 디즈니의 애니메이션 <빅 히어로>의 프레드 Fred는 다음과 같이 설명한다.

“ I wear them front, I wear them back,  
I go inside out, then I go front and  
back!

앞으로 입고, 돌려 입고, 뒤집어서 앞으로  
입고, 뒤집어서 돌려 입고!

”

이를 수학으로 표현해보자.

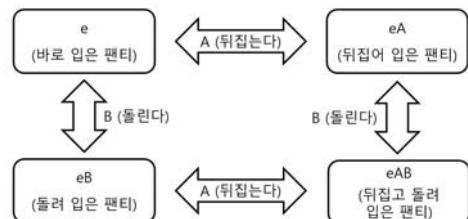
팬티를 바로 입은 상태를 **e**라고 하고, 팬티를 뒤집어 입는 것을 **A**, 돌려 입는 것을 **B**로 표기하자. 예를 들어, 팬티를 뒤집어 입었으면 **eA**, 뒤집은 다음 돌려 입었으면 **eAB** 식으로 표기하는 것이다. 여기서 다음과 같은 사실을 관찰할 수 있다.

**eAA = e** 두 번 뒤집으면 똑같아진다

**eBB = e** 두 번 돌리면 똑같아진다

**eAB = eBA** 뒤집어서 돌리나  
돌려서 뒤집으나 똑같다

여기에 약간의 추론을 더하면 팬티를 입을 수 있는 가능성은  $\{e, eA, eB, eAB\}$  네 가지밖에 없다는 결론이 나온다. 다시 말해 아무리 **e, A, B**를 가지고 표기를 복잡하게 하더라도 결국 저 네 가지 중 하나랑 동일하다는 것이다. 이들의 관계를 그림으로 표현하면 다음 그림과 같다.



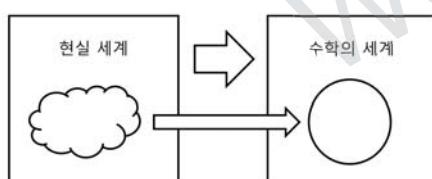
사실 우리는 수학자가 아니더라도, 그리고 프레드처럼 몸소 실천하는 사람이 아니더라도 팬티를 입는 방법이 4가지뿐이라 는 것을 생각해낼 수 있다.

그렇다면 수학자들은 왜 이러는 것일까?

## I. 수학자들은 왜 이러는 걸까

현실 세계를 설명하는 방법은 여러 가지가 있지만, 시대에 따라 사람들이 인정해주는 주류 해석이라는 것이 하나씩 있었다. 고대 사람들은 여러 명의 신들이 태양과 행성을 움직인다고 믿었고, 그 당시에는 아직 팬티도 발명되지 않았다. 수천 년이 지난 지금 사람들은 행성이 움직이는 물리법칙을 알고 있고, 그 위치도 거의 정확하게 예측할 수 있다. 현실 세계를 설명하기 위한 방법 가운데 오늘날 사람들이 인정해주는 가장 좋은 방법은 바로 과학인 것이다.

그런 과학에서는 현실을 분석하기 위해 구체적으로 어떤 방법을 사용할까? 과학자들은 행성의 움직임을 수식과 방정식과 그래프로 표현하고, 팬티를 **eAB**로 표현한다. 이 두 가지에는 공통점이 있다. 바로 현실 세계를 수학의 세계로 옮겨놓고 분석한다는 것이다.



모두 알고 있듯이 현실 세계는 수학의 세계와 많이 다르다. 행성의 궤도는 우리가 몰랐던 소행성 때문에 미세하게나마 틀어질 수도 있고, 팬티 중에서도 구조적으로 뒤집을 수 없는 팬티도 있을 것이다. 하지만 우리는 우리가 알고 싶은 몇 가지 현상을 연구하기 위해 일부러 현실 속의 나머지 성질을 무시하고 수학의 세계로 끌어들이는 것이다. 팬티를 생각해 보자. 우리는 앞서 팬티에 대해 우리도 모르게 몇 가지 가

정을 해버렸다.

- 팬티는 구멍이 2개인 곡면이다.
- 팬티의 재질은 부드러워서 쉽게 뒤집을 수 있다.
- 팬티를 뒤집는 과정에서 늘어날 수는 있어도 찢어지진 않는다.

어떤 사람은 심지어 팬티를 수학적으로 더 엄밀하게 정의했을 테지만, 여기서 그렇게까지 하고 싶지는 않다. 중요한 건 우리가 팬티의 세계를 수학의 세계로 옮기기 위해 여러 가지 세부사항을 희생시켰다는 점이다. 실제 팬티는 좀더 복잡하게 생겼지만 ‘입는 방법’만을 알기 위해 다른 부분을 생략한 것이다.

그렇게 해서 무엇을 얻었을까? 우리는 팬티를 돌리고 뒤집는 것을 위한 표기법과 연산을 만들었다. 그 덕분에 우리는 팬티를 구체적으로 묘사하지 않고 **eAB**라고만 써놔도 어떤 상태인지 알 수 있게 되었다. 또한 우리는 이 연산의 성질을 밝혀냈기 때문에, **eAABBAABBBA**라고 쓰여 있어도 그것이 **eAB**랑 똑같다는 사실을 (시간은 걸리겠지만) 기계적으로 추론해낼 수 있다.

이처럼 팬티의 세계를 수학의 세계로 옮겨놓으면 팬티하고는 전혀 관계없어 보이는 추상적인 연산의 세계가 되어 버린다. 그 결과 팬티를 직접 돌리고 뒤집어 보는 대신 간단한 연산으로 편하게 조작할 수 있어지기도 했고, 팬티만 생각했을 때에는 알 수 없었던 새로운 사실을 알게 되기도 했다. 그 중에서는 거꾸로 팬티를 이해하는데에 도움을 주는 것이 있을지도 모른다.



## II. word2vec

위 그림은 <http://w.elnn.kr/> 의 화면을 캡처한 것이다. 혹시 위와 같은 웹사이트를 본 적이 있다면, word2vec을 알고 있는 것이다. 쉽게 말해서 word2vec은 단어를 더하고 뺄 수 있는 도구다. 이걸로 예를 들어 **서울 : 한국 = 도쿄 : ?** 같은 질문을 풀 수 있다. **한국 - 서울 + 도쿄 = 일본**, 그러므로 답은 **일본**이다.

팬티의 경우처럼 word2vec도 현실 세계를 수학의 세계로 옮겨놓은 결과다. 좀 더 자세히 말하면, 단어 하나하나를 ‘벡터’에 대응시켜서 단어 전체의 공간을 벡터 공간으로 옮겨놓은 것이다. 벡터가 정확히 무엇인지는 여기서 다루지 않겠지만, 두 가지 장점이 있다는 건 말할 필요가 있다. 벡터끼리는 덧셈 · 뺄셈을 쉽게 할 수 있고, 벡터끼리는 거리를 쟈 수 있어서 어떤 게 가깝고 어떤 게 멀리 있는지를 알 수 있다.

생각해 보면 단어들끼리에도 멀고 가까운 거리가 있을 것 같다. ‘사과’는 ‘복숭아’에는 가까이 있지만 ‘오징어’하고는 멀리 떨어져 있을 것이다. 하지만 얼마나 멀리 떨어져 있을까? 구체적인 거리를 예측해보기 위해 word2vec을 이용할 수 있을 것이다. 하지만 덧셈과 뺄셈에는 어떤 의미가 있을

까? ‘인간’에 ‘오징어’를 더하면 어떤 결과가 나올까? 혹시 원빈이 등장하기 전에는 그게 다른 결과가 나왔을까? 쉽지는 않은 문제다.

단어 하나하나는 추상적인 개념에 해당하기 때문에 직접적으로 덧셈 뺄셈을 정의할 수 없는 것들이 훨씬 많을 것이다. 만약 단어 공간이라는 것이 존재한다면, 연관된 단어끼리 서로 얹히고 설킨 아주 정교한 구조를 가질 것이다. 반면 벡터 공간은 수학의 세계 속에서도 아주 단순하고 다루기 쉬운 구조에 속한다. ‘오징어’는 심해생물을 뜻할 수도 있지만 어떤 특징을 가진 사람에 대한 비유가 될 수도 있다. 하지만 이 단어를 벡터 공간으로 옮겨버리면 이런 정보는 사라져버릴 가능성이 크다.

수학의 세계로 옮겨놓으면 무언가를 희생해야 한다는 말을 위에서 했는데, 여기서도 복잡한 단어 공간을 단순하게 만들면서 여러 가지가 희생되었다. 대신 무엇을 얻었을까? 왜 이런 작업을 하는 걸까? 바로 컴퓨터를 위해서다. 컴퓨터가 인간의 언어를 더 이해할 수 있도록 word2vec을 만든 것이다.

## III. 수학의 세계와 컴퓨터의 세계

요즘 컴퓨터는 바둑으로 사람을 이길 수 있다. 하지만 원래 컴퓨터는 구조상 ‘덧셈과 곱셈’ 정도밖에 못한다. 컴퓨터가 덧셈과 곱셈밖에 못한다니, 그럴 리가 없다고? 만약 의심이 된다면 주변의 전문가한테 물어보라. 아니면 트위터에 시비를 거는 방법도 있다. 맞춤법을 한두 군데 틀려주는 것이 좀더 효과적이다.

컴공과한테 들었는데 게임엔진 만들필요 없다며? 원래 cpu에 그런 기능 다있다던데 왜만들ㅋㅋㅋㅋㅋ

운이 좋다면 전공생들이 동시다발적으로 나타나서 전자회로의 구조부터 친절하게 차근차근 설명해줄 것이다. 트위터는 좋은 곳이다.

CPU에 모든 기능이 다 들어 있다면 프로그래머가 필요하지도 않을 것이다. ‘덧셈과 곱셈’을 어떻게 효율적으로 활용하여 현실의 문제를 풀도록 컴퓨터에게 시킬 것인가, 이것이 프로그래머들의 고민이다. 프로그래머는 어떻게 그 고민을 해결할까? 일단 현실 속 문제를 계산 가능한 수학의 문제로 바꾼 다음, 이걸 다시 단순한 컴퓨터가 풀 수 있는 ‘덧셈과 곱셈’의 문제로 바꾼다.



예를 들어보자. 포토샵의 블러blur는 이미지를 흐리게 만드는 효과를 준다. 구글에 ‘포토샵 블러’라고 검색해보면 쉽게 알 것이다. 컴퓨터는 이걸 어떻게 하고 있는 것일까? 각 픽셀의 색깔 값이랑 적당히 근처에 있는 픽셀의 색깔 값이랑 평균을 내면 된다. 붙어있는 픽셀끼리 색깔이 서로 상쇄되기 때문에 이미지가 흐려지는 원리다. 평균은 덧셈과 곱셈을 잘 조합해서 계산할 수 있으므로 컴퓨터가 할 수 있는 계산이다. 포토샵의 다른 여러 가지 기능도 전부 컴퓨터가 할 수 있는 계산으로 구현된 것들이

다.

그렇다면 컴퓨터에게 인간의 언어를 이해시키려면 어떻게 해야 할까? 우선 언어의 공간을 컴퓨터가 계산을 할 수 있는 계산 가능한 공간으로 옮기는 작업이 필요할 것이다. 다른 사람의 말을 이해하는 과정, 그 말을 듣고 적절한 대답을 만들어내는 과정 등을 컴퓨터가 ‘계산’할 수 있도록 프로그래머가 수학 공간으로 끌어와 줘야 한다. 그 시도 중 하나가 word2vec인 것이다. 벡터 공간은 수학적으로 간단한 공간이므로 컴퓨터가 계산하기도 쉽고 프로그래머가 실수할 가능성도 적다. 단어를 이 벡터 공간에 배치하면 각 단어의 의미가 무엇인지 (여러 가지가 생략되어 정확하지는 않겠지만) 컴퓨터가 쉽게 판단할 수 있다는 것이다.

#### IV. 결론

지금까지 현실 세계를 수학의 세계로 옮겨놓는 과정 몇 가지를 보았다. 수학의 세계로 옮겨지면서 복잡한 개념이 몇 개의 기호로만 표현되는 것도 보았고, 복잡한 개념이 컴퓨터로 구현되는 것도 보았다. 하지만 우리가 수학을 이용하는 이유는 이것 말고도 더 있다.

수학의 세계에서는 개념이 잘 정의되어 있고 가정만 충분히 주어져 있으면 무언가를 증명할 수 있다. 그러므로 개념과 상황이 똑같이 주어졌다면 항상 옳은 주장이 있고 항상 틀린 주장이 있다는 것이다. 반면 현실 세계엔 ‘항상 옳은 주장’은 거의 존재하지 않는다. 그러므로 우리는 어떤 주장을 하기 위해 현실을 수학의 세계로 끌어들이

는 것이다.

예를 들어, 사회과학의 많은 분야에서는 통계를 쓴다. 인간의 사회는 복잡하고 불확실하며 무엇보다도 실험실에 집어넣을 수가 없기 때문에 연구하기 힘들다. 반면 통계의 세계에선 확률과 확률 사이의 계산만큼은 확실하고 틀림이 없다. 다만 몇 가지 가정을 해줘야 할 뿐이다. 그렇기 때문에 사회학자들은 인간의 사회를 확률의 세계로 옮기는 과정에서 그 몇 가지 가정을하게 된다. 그들의 연구를 검증하기 위해서는, 그들이 어떤 가정을 했고 어떤 세부사항이 누락되었는지, 데이터 수집을 제대로 했는지를 보면 된다. 가정이 제대로 되어 있다면 통계학과 수학 자체에서 오류가 나올 일은 별로 없을 것이기 때문이다.

장황하게 썼지만 마감이 다가오므로 결론을 내겠다. 많은 사람들이 수학을 어려워하지만, 실제로는 그 반대다. 현실 세계의 문제가 훨씬 더 어렵기 때문에 그것을 쉽게 풀어내려고 만든 도구가 바로 수학이다. 물론 수학자들은 그 수학 자체를 더 엄밀하고 더 정교하게 발전시키고 있지만, 많은 사람들에게 수학은 그냥 도구고, 도구여야 한다. 이 부족한 글로나마 수학을 두려워하는 사람이 조금이나마 줄어들었으면 좋겠다.

## 더 알고 싶은 이를 위해

위키를 통해 공부하는 것은 추천하지 않는다. ○○위키든 위키○○든 마찬가지다. 사전식으로 서술된 정보로 공부하는 데에는 한계가 있기 때문이다. 만약 위키를 통해 하나를 배웠다고 하더라도 그 다음에는 무엇을 공부해야 할지, 어느 방향으로 가야

할지 위기는 알려주지 못한다. 그리고 위키에 아예 틀린 정보가 있는 경우도 많다.

추천하고 싶은 좋은 책이 많지만, 영어 원서나 교과서는 언급하지 않겠다. 그런 책을 혼자서 공부할 수 있는 사람이라면 애초에 이 글에 의지할 필요도 없을 것이다.

이 글의 취지와 가장 가까운 책을 꼽으라면 수학자 김민형, 김태경의 <수학의 수학>(은행나무, 2016)일 것이다. 모든 것은 연산 가능하며 연산 가능한 구조를 탐구하는 것이 수학의 본질이라는 이야기를 주로 한다. 중간중간 수식은 나오지만 건너뛰더라도 대체적인 맥락은 파악할 수 있을 것이다.

유지니아 챕의 <수학을 요리하다>(이화란 역, 처음북스, 2016)은 수학에서 추상화가 왜 중요한지를 요리 같은 생활에 근접한 비유를 들어서 설명한다. 문체가 어떤 이에게는 친절할 수도 어떤 이에게는 장황할 수도 있다.

이 글의 본문과는 직접적인 관련이 없지만, 유카 히로시의 <수학 걸>(김정환 역. 동아일보사, 2008)은 언급할 만 하다. 중고등학교 수학 범위 중 고난이도의 문제풀이를 라이트노벨 형식으로 소개한다. 다시 말해 여고생이 남고생에게 설명해준다는 방식이다.

이미 중고등학교 수학까지는 익숙한 사람이라면 티모시 가워스의 <아주 짧게 소개하는 수학>(박기현 역, 교우사, 2013)도 추천한다. 수학의 논증 과정이 어떠한지, 역사적으로 논리적 오류가 있었는지 등을 제목처럼 짧게 소개한다. 영어를 직역한 문체가 익숙하지 않다면 읽는 게 다소 느려질

수도 있다.

클리퍼드 피오버의 <수학의 파노라마>(김지선 역, 사이언스북스, 2015)는 내용도 괜찮지만, 혹시 사놓고 읽지 않더라도 장식용으로 훌륭한 기능을 한다. 수학의 중요한 개념들을 소개하면서 알록달록한 사진을 실어서 마치 화보처럼 꾸며놓은 책이기 때문이다. sscC 1st

sscc 1st  
Web ver.

SSCC '1st'

앞으로도 계약되길!!

- R&D9



SSCC 1st  
Web ver.

# Authors

of SSCC 1st

## Akey

아홉꼬리교단

twt @AkeyArqriser

민달팽이의 교미 같이 농후한 Author

“ Akey입니다. 우선 SSCC에 투고하게 되어 영광이고, 이런 기념비적인 회지에 글을 실을 수 있도록 자리를 마련해주신 디버거님께 감사드립니다. 서브컬쳐 관련 학술논문집이라니 막 두근두근하지 않나요? 덕분에 재미있게 작업할 수 있었습니다.

메인 주제는 “생물학적 동성 간 아이만들기” 입니다. 좀 더 직설적인 말을 쓰고 싶었지만 그랬다면 이 회지가 레드존에 들어가는 불상사가 생기는 관계로 최대한 순화해서 썼고요. 제목의 유래는 다들 아시리라 믿습니다. 많은 분들이 전산학 쪽 페이퍼를 내실 거라 생각됩니다마는 #전공을\_살려\_유익한\_정보를\_말해보자 같은 걸 한다 치면 저는 굶어죽어도 이상하지 않을 이학계열이라 생물학 관련하여 글을 쓰게 되었습니다. 부디 재미있게 읽어주시면 감사하겠습니다.

다음 회지는 좀 더 캐주얼한 것으로 가 보겠습니다. 술 만드는 방법이라던가 이것저것 많이 생각하고 있으니, 기회가 되면 SSCC 2호에서 뵙도록 하겠습니다.

## 봄스

한국 서브컬쳐 작품 속의 ‘한국적임’을 찾아서 Author

twt @\_bom bomss 인생이 심란할 때 자살 대신 가슴을 외치는 것으로 마음을 다스리고 있음.

“ 편집 도와준 룸메이트님 고맙습니다. 사랑해...원고 늦어서 죄송합니다 편집자님...그리고 바이클론즈 꼭 봐주세요 정말 좋은 작품입니다. 유튜브에서 전편 무료 보기 할 수 있어요...제가 잘 해드릴게요 꼭 봐주세요....

## sapphire

sapphire.sh

s@pphi.re

아이돌마스터 신데렐라 걸즈 스타라이트 스테이지 이미지 리소스 분석하기 Author

twt @sapphire\_dev

본업은 프로그래머, 부업은 프로듀서입니다.

“ 정형화된 아티클을 작성하는 게 너무 오랜만이라 부족한 점이 많은 것 같네요. 아무쪼록 잘 부탁드립니다.

## KeV

twt @\_NThalpy

Face Detection of Monoeye Character Author

이미지 프로세싱과 형식언어에 관심이 있는 사람입니다. 관심만 있고 실력은 없군요 하하.

“ 제가 제일 좋아하는 서브컬쳐와 평소에 관심이 있던 분야를 붙여봤습니다!

## 제이미|theeluwin

theeluwin@gmail.com

CRF를 활용한 한글로 적힌 외래어 및 외국어 받아쓰기 인식 Author

twt @theeluwin

제이미라고 해요- 취미로 CSE 연구 및 개발 하는 귀여운 사람입니다.

트위터에 주로 서식해요

“ 평소에도 서브컬처를 주제로 연구를 해보고 싶다는 생각을 자주 했는데, 마침 SIG SC를 구성하여 이런 기회를 제공해주신 Proceedings of SSSC 1st 편집 대표자 디버거님에게 감사의 말씀을 올립니다. Special thanks to 서울대학교 IDS 연구실의 최지현군.

## Otter

지리산 프로듀서 협회

a19641sk@gmail.com

란코처럼 말하는 트위터 봇 만들기 Author

twt @a19641sk

초보 개발자입니다. 최근, 옆집 지리산 반달가슴곰들이 팬라이트를 흔들며 콜을 넣는 소리가 들려서 무섭습니다.

“ 최대한 쉽게 쓰려고 노력했습니다. 제 글이 가장 쉬울 것이란 생각을 나라도 쉬운 것을 쓰자는 소명의식으로 바꾸지 않았다면 영원히 제출하지 못했을 것 같습니다.

## Chœ Byung-yoon

KAIST HAJE

byungyoonc@gmail.com

Wrapping Low-level Graphics Library for Author

twt @kevincby\_

Interactive Application

Proceedings of SSSC 1st Design Cooperation

하는 건 없는데 고통받는 전산과 CG랩 대학원생입니다.

게임 하는 것과 만드는 걸 좋아해요.

“ 우선 읽어주셔서 감사합니다. 이런 형식의 튜토리얼 텍스트를 적어보는 것은 처음이라. 거기다 시간을 많이 두고 쓴 글이 아니어서 획설수설한 감이 있군요. 특히 한국어로는요! 서투른 글입니다만 모쪼록 누군가에게는 조금이나마 도움이 됐으면 좋겠습니다. 이런 글을 쓸 수 있는 기회를 주신 SIG SC에게 많은 감사를 드리고, 앞으로도 많은 발전이 있기를 바랍니다!

## Mappy The Kat

twt @Job\_Kat

대신귀여운평셔널을드리겠습니다 Author

C#성애자인 잡고양이입니다

“ 만화는 처음이라 이것저것 삽질을 많이 했네요 C++양 이야기도 넣고 싶었지만 여백이 부족하여(...) 넣지 못했습니다 백합물을 만들겠다고 자신만만하게 선언했는데 어디로 가버리고 로리핥는 변태여고생만 남았네요 부족한 작화지만 봐주셔서 정말 감사합니다 왜 학습만화는 산으로 가는가를 절실히 깨달았습니다..(..)

## Ria Fox

국제 매력적인 여우 재단

twt @Ria\_Fox

주토피아는 실재하는가 Author

곡도 쓰고 코딩도 하고 그림도 그리는 공돌이입니다. 여우가 좋습니다.

“ 처음 SSCC를 연다는 말을 들었을 때 예전부터 생각해 온 수인 관련 글을 쓸 생각에 되게 설렜습니다. 마침 인류학과 인류사 관련 책을 잔뜩 읽게 되어서 그걸 레퍼런스 삼아 쓰면 되겠지, 했지만 역시 인문학은 어렵군요 ㅠㅠ 글을 쓰면서도 '아..... 그냥 3D 모델 꼬리 리깅같은거나 쓸걸 ㅠㅠㅠ' 했지만, 다 쓰고 보니까 부족한 글이라도 굉장히 뿌듯합니다. 굉장히 심심하고 재미없는 글이지만, 읽어 주셔서 감사합니다. 그리고 회지 발간에 많은 수고를 해 주시고, 이 글을 쓸 기회를 주신 디버거님께도 정말로 감사드립니다.

## 小栗戸栗

강화기

twt @ogri\_togri

○○위키가 알려주지 않는 수학 이야기 Author

평타공격시 레드락 발사

“ 2단 구성이라 수능 시험지처럼 보인다. 이 때 글쓴이의 심정으로 알맞는 것은?

## SSCC 1st Committee

### Debugger

잉여력 학회

debugger0net@gmail.com

Proceedings of SSCC 1st Editor in Chief

twt @debuggerD

SIG SC에서, 하라는 집필은 안 하고!, 편집을 담당하고 있습니다.

“ 참여해 주신 여러분, 감사합니다! 여기까지 읽어주신 독자 여러분께도 감사드려요!! ~☆

## Cover Image Credit

아루

내 방 침대

twt @aru\_soc

Proceedings of SSCC 1st Cover Illustration

그림을 그리는데 그림을 못 그립니다.

“안녕하세요 아루입니다. 디버거님의 반강제적인(?) 권유로 sscc의 기념비적인 창간호 표지를 맡는 영광을 얻게 되었습니다. 표지의 그림은, 까페에서 프로그래밍을 하다 막혀서 고민하는 귀여운 개발자의 모습을 그려보았습니다. 책의 얼굴이 되는 표지인데 괜찮은 그림을 그렸을지 걱정이 되는군요. 다음 호가 있다면 그 때는 아티클로 참가할 수 있다면 좋겠습니다. 이 책이 나올 때까지 많은 노력을 하신 디버거님과 그 외 필자님들에게 감사를 표합니다. 그리고 독자님에게도 책을 구입해주셔서, 그리고 여기까지 읽어주심에 감사를 드립니다.”

축전을 보내주신

바다코코볼

님께, 감사의 말씀을 드립니다.

SIG SC

Copyright © 2016-2017 SIG SC Conference 1st Authors.