

# ● 포트폴리오

전민수



# 전민수

1990.06.19

단국대학교 컴퓨터과학 학사 과수석 졸업(2016.02)

프로그래밍이라면 가릴 것 없이 다 좋아하는  
뻘속까지 프로그래머

GitHub URL: <https://github.com/codejun6/university-portfolio>

Contact: [codejun6@gmail.com](mailto:codejun6@gmail.com)



## Contents

### 1. '손으로 움직이는 세상' 프로젝트

(C++ 기반 오픈소스 OpenCV를 이용한 손바닥 패턴 인식)

### 2. 'Fast Shopper' 프로젝트

(오픈소스 Zbar를 이용한 바코드 인식 안드로이드 앱과 간단한 Java 기반 서버의 TCP 통신)

### 3. 'Mini-CPL 인터프리터' 프로젝트

(간단한 수식 계산용 언어와 해당 언어의 인터프리터)

### 4. 'Cocoa-Uinx 기반 채팅 시스템' 프로젝트

(Cocoa 기반 클라이언트와 C 기반 서버의 TCP 통신)

### 5. '사진 관리 프로그램 갈무리' 프로젝트

(이미지 파일 관리 및 편집용 MFC 프로그램)

### 6. Z사 iOS 개발 테스트

(iOS 개발 능력 테스트를 위해 제작한 iOS 앱)

# 1

## '손으로 움직이는 세상' 프로젝트

- 1-1. 프로젝트 소개
- 1-2. 구현과정과 후기
- 1-3. 실행화면

※ GitHub 내 소스위치: 'university-portfolio' repository -> HandCotrollInterface

## 1-1. '손으로 움직이는 세상' 프로젝트 소개

### 프로젝트 소개

이 프로젝트는 대학교 2학년 때 본교 대학원 ACDM 연구실에서 팀을 꾸려 한이음 IT멘토링으로 진행한 프로젝트입니다.

'손으로 움직이는 세상' 프로젝트는 마우스나 키보드와 같은 하드웨어가 없을 때, 저가형 웹캠 하나로 모두 제어할 수 있는 인터페이스를 만들자는 생각으로 계획했습니다.

손동작을 통해 마우스 커서를 제어하는 손동작 인식 인터페이스 프로그램과 인터페이스의 효과를 보여주기 위한 WPF 기반 콘텐츠 프로그램으로 구성했습니다. 콘텐츠 프로그램은 인터페이스의 효과를 보여주기 위해 간단한 게임과 그림판 콘텐츠로 개발했습니다.

### 개발기간 및 인원

#### 개발기간

2010.06.01  
~ 2010.12.31

#### 개발인원: 5명

#### 담당부분 (기여도 60%)

적외선 발광기 제작  
및 웹캠 개조,  
인터페이스 관련  
모든 개발 업무  
(WPF 기반 콘텐츠  
프로그램 제외)

### 개발환경

#### 운영체제

Windows XP

#### 개발언어

C++, C# (WPF)

#### 개발도구

VS 6.0, VS 2008

#### 오픈소스

OpenCV 1.0

#### 기타

Windows 기반 PC,  
적외선 인식 웹캠,  
적외선 발광기

## 1-2. '손으로 움직이는 세상' 구현과정과 후기

### 구현과정

손동작 인식 인터페이스는 웹캠을 통해 입력된 실시간 영상을 영상처리를 함으로써 손동작을 인식하는 프로그램입니다. 처음 계획은 일반 웹캠을 통해 손동작을 인식함으로써 PC와 상호작용을 하려 했으나 제한이 많다는 생각이 들었습니다. 그래서 웹캠을 개조해 적외선을 인식할 수 있도록 하여 적외선 발광기와 같이 사용해 손동작을 인식했습니다.

웹캠을 통해 입력된 영상을 C++ 기반 OpenCV 오픈소스를 이용해 영상처리를 했습니다. 적외선 인식 웹캠이기 때문에 웹캠에 가까울수록 영상이 선명하게 찍힐 수 있습니다. 그래서 손을 웹캠에 근접해 움직이도록 설계함으로써 수월하게 인식할 수 있었습니다. 인식한 손바닥을 다루기 쉽도록 흑백으로 이진화한 뒤, 손바닥의 특징을 이용해 손가락의 접힘 여부를 판단했습니다. 이를 이용해 손동작을 분류하고 그에 따라 마우스의 좌클릭이나 우클릭에 매칭하여 인터페이스 역할을 하도록 했습니다.

### 후기

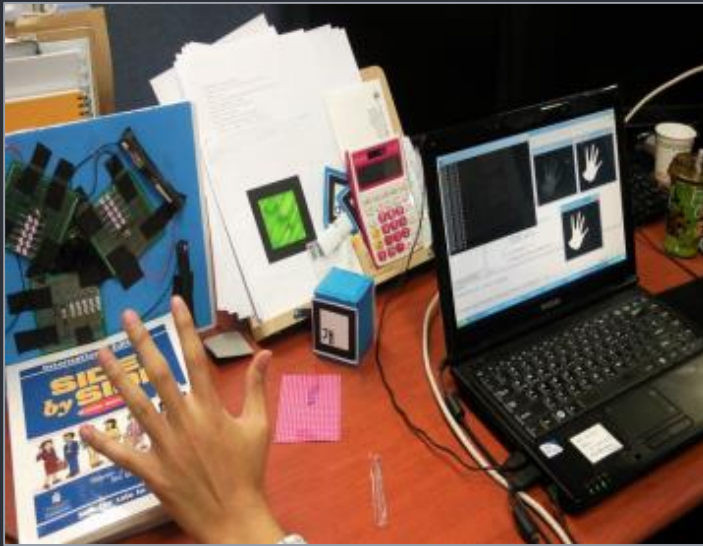
첫 팀 프로젝트였기 때문인지 시행착오가 많았던 프로젝트였습니다. 특히, 다양한 논문을 읽고 좋은 해결책을 찾았지만, 해당 논문의 방법을 직접 구현하는 데 어려움이 많아 구현하지 못한 것이 큰 아쉬움으로 남습니다.

또한, OpenCV 오픈소스를 잘 사용하기 위해 소스코드를 분석하고 테스트하는 데 많은 시간을 할애해 프로젝트의 완성이 지연된 결과를 낳았습니다.

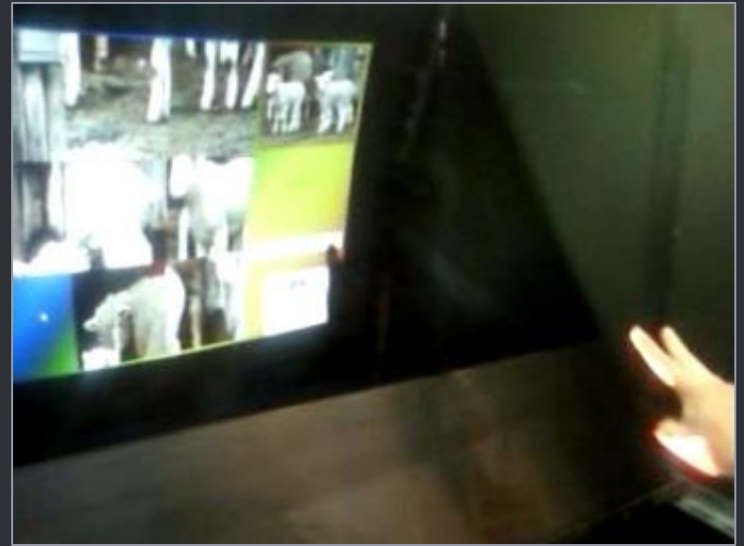
마지막으로 아쉬웠던 것은 혼자서 인터페이스에 대한 개발을 했기 때문에 제대로 된 팀 프로젝트를 경험하지 못한 것이 아쉬웠습니다.

수많은 시행착오로 최종 결과물은 완성도가 높지 않았습니다. 하지만, 그 과정에서 많은 실패를 경험하면서 많은 것을 터득했습니다. 영상처리, 적외선, OpenCV 등에 대한 개념과 많은 논문을 접할 수 있는 좋은 기회였습니다.

### 1-3. '손으로 움직이는 세상' 실행화면



적외선 발광기와 개조한 웹캠을 통해  
손동작 인식을 테스트하는 모습입니다.



시연을 위한 컨테이너를 설치하고, 간단  
한 퍼즐 게임을 시연하는 모습입니다.

## 2

# 'Fast Shopper' 프로젝트

- 2-1. 프로젝트 소개
- 2-2. 구현과정과 후기
- 2-3. 실행화면

※ GitHub 내 소스위치: 'university-portfolio' repository -> FastShopper



## 2-1. 'Fast Shopper' 프로젝트 소개

### 프로젝트 소개

이 프로젝트는 대학교 휴학 시기에 교내 캡스톤 디자인 경진대회를 준비하며 진행한 프로젝트입니다.

Fast Shopper 서비스는 마트에서 빠른 거래가 이뤄지게 함으로써 줄이 길어지지 않도록 도와주는 서비스입니다. 앱을 통해 상품의 바코드를 찍어 각종 정보를 살펴보거나, 장바구니에 등록할 수 있습니다. 마지막 결제 시 장바구니의 결제 버튼을 눌러 요청을 하면, POS 시스템(서버)을 통해 판매원이 확인 후 바로 결제가 완료되게 합니다.

의도한 서비스는 마트의 POS 시스템과 서버를 연동해 결제를 확인할 수 있도록 하는 것이었으나, 결제 서버의 구현은 주가 아니기 때문에 서버는 간단하게 Java Swing, TCP 통신, 파일 DB를 이용해 가상의 결제 확인 과정을 진행하도록 만들었습니다.

### 개발기간 및 인원

#### 개발기간

2013.09.12  
~ 2013.11.14

#### 개발인원: 5명

#### 담당부분 (기여도 100%)

모든 개발 업무  
(안드로이드 앱 및  
Java Swing 기반  
서버 개발)

### 개발환경

#### 운영체제

Windows 8

#### 개발언어

Java (Android)

#### 개발도구

JDK 1.7, Eclipse

#### 오픈소스

Zbar (Bar code  
reader)

#### 기타

안드로이드 기반  
스마트폰, Windows  
기반 PC

## 2-2. 'Fast Shopper' 구현과정과 후기

### 구현과정

Fast Shopper 서비스의 핵심은 클라이언트 앱입니다. 고객이 Fast Shopper 앱을 통해 상품의 바코드를 찍어 장바구니에 모아 놓고, 한 번의 클릭만으로 미리 등록한 카드 정보를 통해 결제를 진행할 수 있습니다. 판매원은 요청 받은 결제 정보와 실제 상품 구성의 일치 여부만 확인하면 되므로 빠른 거래가 이뤄질 수 있습니다.

상품의 바코드를 인식하기 위해 안드로이드용 Zbar 오픈소스를 이용했습니다. 인식한 바코드의 일련 번호는 인식을 할 때마다 서버에 등록된 상품의 바코드와 비교하여 재고가 있는 상품인지 확인합니다. 재고가 있다면 장바구니에 상품이 등록되고 원하는 수량을 조절할 수 있습니다. 쇼핑을 마치고 나면 거래를 할 때, 결제 버튼을 눌러 장바구니에 담긴 상품에 대한 결제 정보를 서버로 전송하고 서버에서 결제 확인이 되면 최종 결제가 완료됩니다.

Fast Shopper 서버는 주가 아니므로 유사한 환경을 조성하도록 단순하게 구현했습니다. TCP 통신을 통해 클라이언트 앱의 요청에 대해 응답을 하고, 상품의 바코드는 바코드 리더기로 찍어 파일에 보관했습니다.

### 후기

처음 주어진 시간이 한 달 정도뿐이었고, 안드로이드를 처음 접해서 시간의 압박을 많이 받았습니다. 또한, 혼자서 모든 개발을 맡아야 했기 때문에 부담이 많았던 프로젝트입니다. 하지만, 그로 인해 짧은 시간 안에 많은 것을 배울 수 있었습니다.

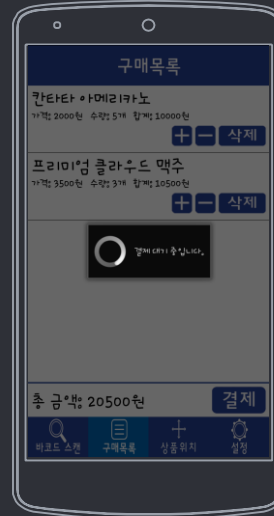
일단, 안드로이드의 구조와 개념, 특징 등을 빠르게 익혀 안드로이드에 입문한 좋은 기회였습니다. 또한, 어떻게 좋은 서버 구조를 만들 수 있을지 고민했습니다. 이 프로젝트에서는 요청과 응답 메시지의 앞 부분에 메시지의 종류를 첨부해 메시지를 구별하는 메시지 기반 구조로 서버를 구현했습니다. 마지막으로 앱은 어떤 프로그램보다 디자인과 인터페이스가 중요하다는 것을 깨달았습니다.

아쉬운 점은 더 많은 기능을 넣지 않아 다양한 활용을 보여주지 못한 점과 결제 서버를 너무 단순하게 구현해 다른 기술을 적용해 보지 못한 점이 아쉽습니다.

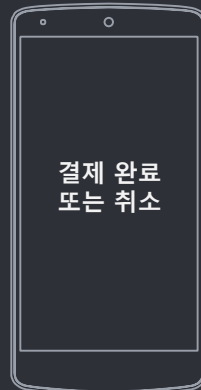
## 2-3. 'Fast Shopper' 실행화면



바코드 스캔 및  
구매목록 결정



결제 요청



구매목록 확인 후,  
구매 확인 또는 취소



# 3

## 'Mini-CPL 인터프리터' 프로젝트

- 3-1. 프로젝트 소개
- 3-2. 구현과정과 후기
- 3-3. 실행화면

※ GitHub 내 소스위치: 'university-portfolio' repository -> Mini-CPLInterpreter

### 3-1. 'Mini-CPL 인터프리터' 프로젝트 소개

#### 프로젝트 소개

이 프로젝트는 대학교 4학년 때 컴파일러 강의를 들으면서 기말 과제로 진행한 프로젝트입니다.

이 과제의 사전 과제로 강의 교재의 저자가 간단한 프로그래밍 언어로 구현한 BBI 인터프리터를 분석했습니다. 이 과제는 분석한 내용을 토대로 변수 할당과 계산이 가능한 새로운 언어를 정의하는 과제였습니다.

Mini-CPL 인터프리터는 변수 사용이 가능한 간단한 계산 언어인 Mini-CPL(Calculation Programming Language)을 해석해 실행하는 프로그램입니다. 과제로 주어진 규칙에 따라 언어를 정의하고, 그 언어를 인식할 수 있는 인터프리터를 구현했습니다.

#### 개발기간 및 인원

##### 개발기간

2015.05.22

~ 2015.05.27

개발인원: 1명

#### 개발환경

##### 운영체제

Windows 8

##### 개발언어

C++ (C 위주)

##### 개발도구

VS 2008

## 3-2. 'Mini-CPL 인터프리터' 구현과정과 후기

### 구현과정

Mini-CPL(이하 CPL)은 변수의 사용이 가능한 간단한 계산용 프로그래밍 언어이며, Mini-CPL 인터프리터는 CPL로 이뤄진 소스 코드를 바로 인식하여 실행하는 인터프리터입니다. 이 인터프리터는 간단한 수식의 형태만 인식할 수 있는 형태로 개발되었으므로 실용성보다 학습의 측면에서 진행하게 되었습니다.

이 인터프리터는 기본적인 인터프리터의 구조를 따르고 있습니다. 소스 코드를 입력하면 반복적인 어휘 분석을 통해 토큰을 취하고, 구문 분석을 통해 중간 코드를 만들어 냅니다. 중간 코드는 역할의 분리, 성능 향상, 확장성 부여 등을 고려하여 생성했습니다. 최종적으로 중간 코드를 읽어 들이며 실행을 하게 됩니다. 어휘 분석, 구문 분석, 실행 중 어느 곳에서라도 에러가 발생하면 에러를 출력하고 중지하도록 했습니다.

어휘 분석을 위해 키워드 테이블을 준비하고, 그에 대응하는 키워드와 변수, 상수에 대응하는 주소를 심볼 테이블에 할당함으로써 중간 코드를 생성합니다. 또한, 변수와 상수를 유지하기 위해 데이터 메모리와 중간 코드 보관을 위한 벡터를 준비해 구현했습니다.

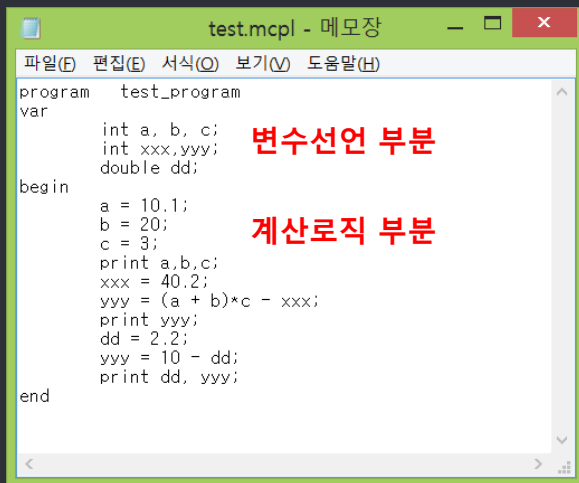
### 후기

기존에 했던 프로젝트들과는 다른 느낌의 프로젝트여서 아주 즐거운 기분으로 진행했습니다. 소스 코드를 읽어가며 해석함으로써 새로운 언어를 창출할 수 있다는 게 큰 매력으로 다가왔습니다.

또한, 프로그래밍 언어를 바라보는 시점에 큰 영향을 받았습니다. 예전에는 소스 코드를 봐도 문법이 맞는지 여부만 생각했지만, 이제는 해당 소스 코드가 어떤 원리로 컴파일되거나 인터프리트되는지 고민하게 되었습니다. 그리고 그 안에서 소소한 원리를 깨닫는 재미를 느꼈습니다.

프로젝트를 끝내고 아쉬웠던 점은 더 재미있는 문법이나 기능들을 확장하지 않았다는 것입니다. 이 프로젝트 이후에 다른 인터프리터에 기능을 확장하는 프로젝트를 했습니다. 그 프로젝트에서 어느 부분을 수정함으로써 확장하기 좋은지 배울 수 있었습니다. 배운 것을 토대로 추후 과제에서는 CPL의 기능을 확장하겠습니다.

### 3-3. 'Mini-CPL 인터프리터' 실행화면

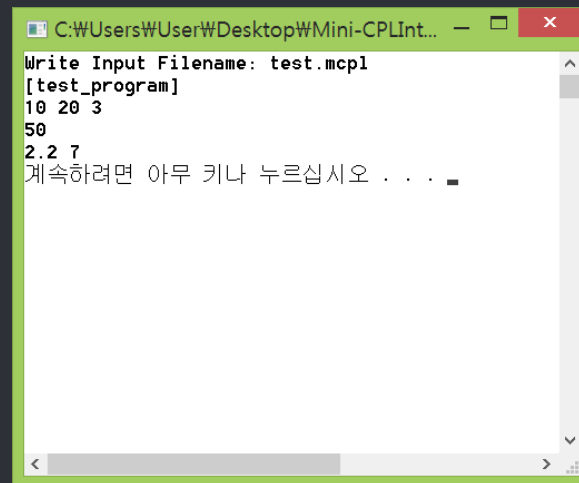


```
program test_program
var
    int a, b, c;
    int xxx,yyy;
    double dd;
begin
    a = 10.1;
    b = 20;
    c = 3;
    print a,b,c;
    xxx = 40.2;
    yyy = (a + b)*c - xxx;
    print yyy;
    dd = 2.2;
    yyy = 10 - dd;
    print dd, yyy;
end
```

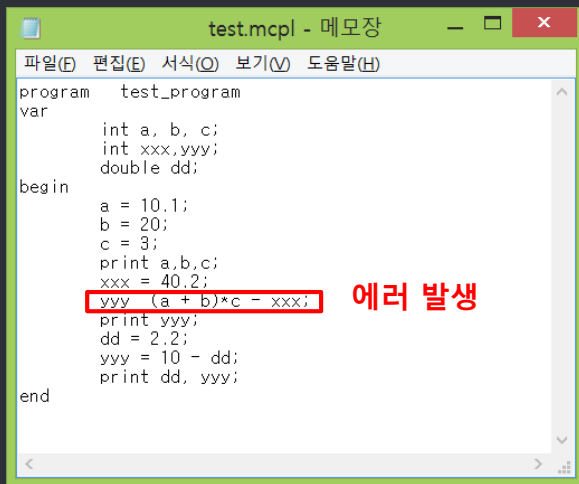
변수선언 부분

계산로직 부분

변수선언 부분과 계산  
로직 부분으로 나누어  
처리하며, print 명령을  
통해 변수 및 상수를  
출력한다.



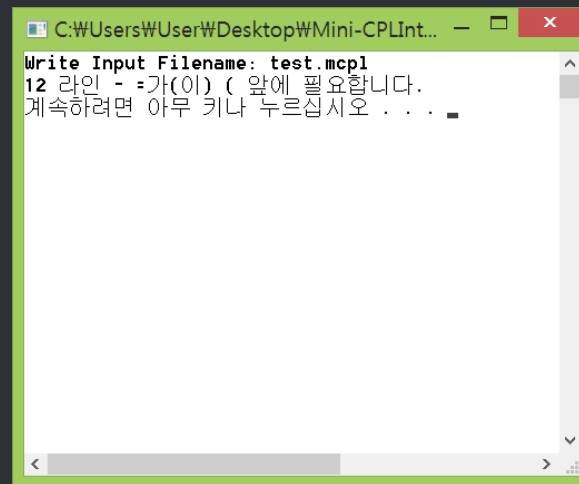
```
Write Input Filename: test.mcpl
[test_program]
10 20 3
50
2.2 7
계속하려면 아무 키나 누르십시오 . . .
```



```
program test_program
var
    int a, b, c;
    int xxx,yyy;
    double dd;
begin
    a = 10.1;
    b = 20;
    c = 3;
    print a,b,c;
    xxx = 40.2;
    yyy = (a + b)*c - xxx;
    print yyy;
    dd = 2.2;
    yyy = 10 - dd;
    print dd, yyy;
end
```

에러 발생

대상 소스코드에 에러  
가 있는 경우, 어떤 에  
러인지 알려주고 프로  
그램을 종료한다.



```
Write Input Filename: test.mcpl
12 라인 - =가(이) ( 앞에 필요합니다.
계속하려면 아무 키나 누르십시오 . . .
```

## 4

# 'Cocoa-Ux 기반 채팅 시스템' 프로젝트

- 4-1. 프로젝트 소개
- 4-2. 구현과정과 후기
- 4-3. 실행화면

※ GitHub 내 소스위치: 'university-portfolio' repository -> SimpleChatting



## 4-1. 'Cocoa-Unix 기반 채팅 시스템' 프로젝트 소개

### 프로젝트 소개

이 프로젝트는 대학교 4학년 때 네트워크 프로그래밍 강의를 들으면서 기말 과제로 진행한 프로젝트입니다.

이 채팅 시스템은 Cocoa 프레임워크 기반 클라이언트와 Unix C 기반 서버로 구성됩니다. 채팅 시스템의 기본 기능으로 로그인 기능, 방 채팅 기능이 있으며, 방을 개설하고 들어가게 되면 해당 방에 속한 사용자들끼리만 대화를 할 수 있습니다. 그 밖의 기능으로는 화상채팅, 귓속말, 파일전송 등을 추가하려 했지만 시간 관계상 추가하지 못했습니다.

제가 구현한 채팅 시스템은 기능은 적지만, 안정성에 초점을 맞춰 개발했습니다. 타 학생들이 많은 기능에 초점을 맞춰 기본 기능마저 버그가 많아 서버가 죽어버리는 것이 대부분인 반면, 저는 기본 기능만은 절대 죽지 않게 개발했습니다.

### 개발기간 및 인원

#### 개발기간

2015.05.11  
~ 2015.06.14

개발인원: 1명

### 개발환경

#### 운영체제

OS X Yosemite

#### 개발언어

Objective-C (Client),  
C (Server)

#### 개발도구

Xcode, GCC 4.2.1

#### 기타

OS X 기반 PC

## 4-2. 'Cocoa-Ux 기반 채팅 시스템' 구현과정과 후기

### 구현과정

이 채팅 시스템의 클라이언트 부분은 OS X의 Cocoa 프레임워크로, 서버 부분은 Unix 기반 C로 개발했습니다. 클라이언트를 개발할 때 네트워크 관련 기능은 서버의 소스 코드를 재사용할 수 있었습니다. 따라서 클라이언트는 Objective-C와 C를 결합해 사용했습니다.

클라이언트를 구현할 때 가장 힘들었던 부분은 GUI였습니다. 이 프로젝트를 통해 Objective-C를 처음 접하면서 많은 문제가 발생했습니다. 최종적으로는 원하는 디자인으로 GUI를 구현할 수 있었습니다.

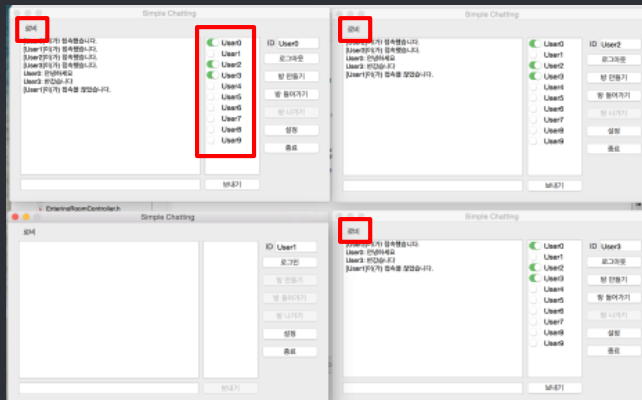
서버는 채팅 시스템의 핵심으로 서버가 죽으면 채팅 시스템은 동작할 수가 없습니다. 그래서 저는 서버만은 안정성이 좋게 만들자는 목표를 두고 구현을 했습니다. 클라이언트와 서버 간의 통신은 TCP 통신 기반이며, 주고 받는 메시지의 포맷은 앞 4바이트에 메시지 종류를 할당하고 그에 따라 뒤의 구조가 결정되도록 구성했습니다. 편의를 위해 이미 가입된 아이디 10개를 준비했으며, 서버의 메인 쓰레드에서는 로그인을 판정합니다. 로그인이 되면 해당 사용자와 통신하기 위한 쓰레드를 실행하고, 로비에 해당하는 쓰레드에 사용자를 등록합니다. 방을 개설하고 들어가면, 해당 방의 쓰레드에 사용자를 등록합니다. 각 로비 및 방 쓰레드는 자신에 속한 사용자에게만 채팅 내용을 전송함으로써 방 채팅 기능을 이용할 수 있습니다.

### 후기

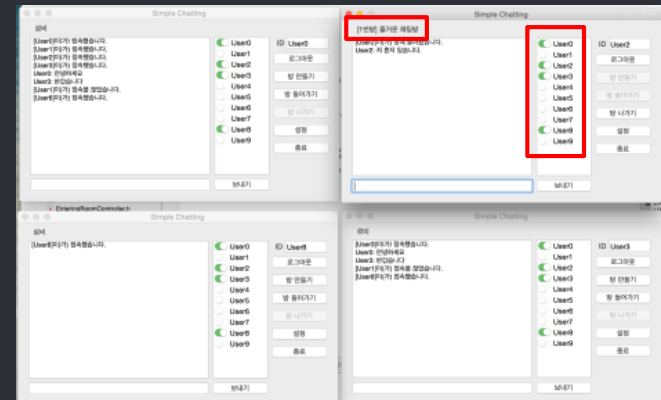
목표 기능을 다 구현하지 못한 가장 큰 이유는 시간 분배의 실패인 것 같습니다. 이 프로젝트의 가장 큰 목적은 기능의 구현이지 GUI는 중요한 부분이 아니었습니다. GUI 구현에 상당한 시간을 잡아먹어 이 시스템의 핵심인 채팅 기능을 구현하는 데 많은 시간을 할애하지 못했습니다. 우선 순위를 잘 따져 시간을 분배하는 경험을 더 쌓아야겠다는 생각이 들었습니다.

이 프로젝트를 진행하며 Objective-C의 매력을 느꼈고, 서버의 구조와 프로토콜을 고민하며 직접 설계할 기회를 가졌습니다. 또한, 이론으로만 배웠던 상호 배제를 뮤텁스를 사용함으로써 적용해볼 수 있었습니다. 단순한 채팅 시스템이 아닌 안정적인 채팅 시스템을 만드는 데 집중하니 고민할 것이 많아 배울 점이 많은 프로젝트였습니다.

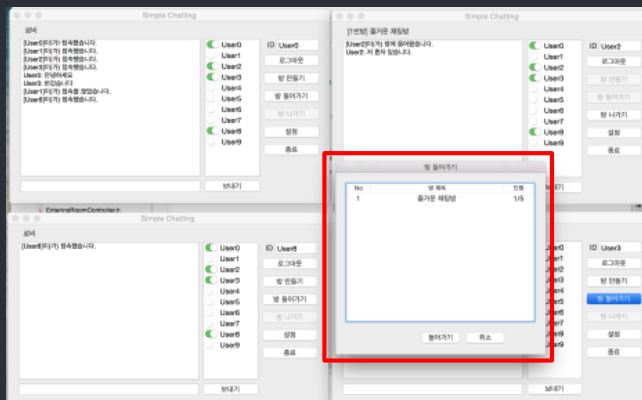
## 4-3. 'Cocoa-Unix 기반 채팅 시스템' 실행화면



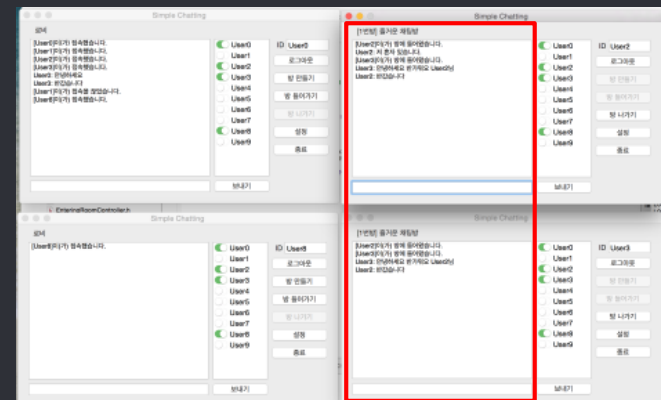
각 사용자가 독립적으로 로그인 및 로그아웃을 할 수 있으며, 로비가 존재합니다.



각 유저는 그룹 대화를 하기 위해 방을 만들 수 있습니다. 유저 목록은 채팅 서버에 접속한 목록으로 유지됩니다.



각 유저는 원하는 방을 선택해 그룹 대화에 참여할 수 있습니다. 방장은 제한인원을 설정할 수 있습니다.



각 방의 그룹 대화는 해당 방에 없는 유저가 확인할 수 없습니다.

## 5

# '사진 관리 프로그램 갈무리' 프로젝트

- 5-1. 프로젝트 소개
- 5-2. 구현과정과 후기
- 5-3. 실행화면

※ GitHub 내 소스위치: 'university-portfolio' repository -> PhotoGalmuri

## 5-1. '사진 관리 프로그램 갈무리' 프로젝트 소개

### 프로젝트 소개

이 프로젝트는 대학교 졸업작품으로 진행한 프로젝트입니다.

처음 계획은 이미지 파일에서 특징 정보(직접 입력한 태그, 밝기 정보, 색상 분포 등)를 추출해 새로운 파일 포맷의 파일을 만들어 사진 관리를 용이하게 하는 라이브러리를 만들려는 의도였습니다. 하지만 시간이 부족해 **사진 관리만 하는 프로그램**으로 탈바꿈 했습니다.

MFC 기반 프로그램으로 여러 경로에 있는 이미지 파일을 한 프로그램에서 관리할 수 있습니다. 이미지 파일로부터 특징 정보를 추출하거나 직접 태그를 입력해 사진 관리(정렬, 검색)를 용이하게 해주고, 간단한 편집 기능 또한 추가했습니다.

### 개발기간 및 인원

#### 개발기간

2015.07.24  
~ 2015.10.25

개발인원: 4명

#### 담당부분 (기여도 70%)

편집 기능 일부,  
검색 및 정렬 기능,  
GUI 구현.

개발 총괄 (통합  
작업 및 관리)

### 개발환경

#### 운영체제

Windows 8

#### 개발언어

C++ (MFC)

#### 개발도구

VS 2010

#### 기타

Windows 기반 PC

## 5-2. '사진 관리 프로그램 갈무리' 구현과정과 후기

### 구현과정

'갈무리'는 C++ 기반 MFC를 이용해 만들었습니다. MFC의 문서-뷰 구조를 적극 활용해보기 위해 SDI 프로젝트로 진행했으며, 영상처리를 위한 이미지 관리용 데이터 타입으로는 GDI+의 CImage 클래스를 이용했습니다. 이미지로부터 원하는 정보를 추출하거나 변형시키는 것은 CImage 객체 내의 Raw 데이터를 이용했으므로 해당 클래스는 이미지를 편리하게 입출력하기 위한 용도로 사용했습니다.

이미지 특징 정보 추출은 이미지를 처음 불러오는 시점에 수행하도록 했으며, 이후 이미지 편집 상황에 따라 변하게 만들었습니다. 이미지 편집은 임시 이미지를 만들어 적용을 해본 뒤, 최종적으로 기존 이미지를 수정하거나 새로운 이미지 파일을 만드는 방향으로 만들었습니다.

프로그램을 종료하고 다시 시작했을 때, 기존 목록을 유지하기 위해 파일 형태의 DB를 이용했습니다. 프로그램을 처음 실행한 것이라면 프로그램과 같은 폴더에 DB 파일을 만들어냅니다. DB에는 각 이미지 파일의 특징 정보와 경로를 모두 저장함으로써 프로그램 재시작 시 실행이 지연되는 것을 막았습니다.

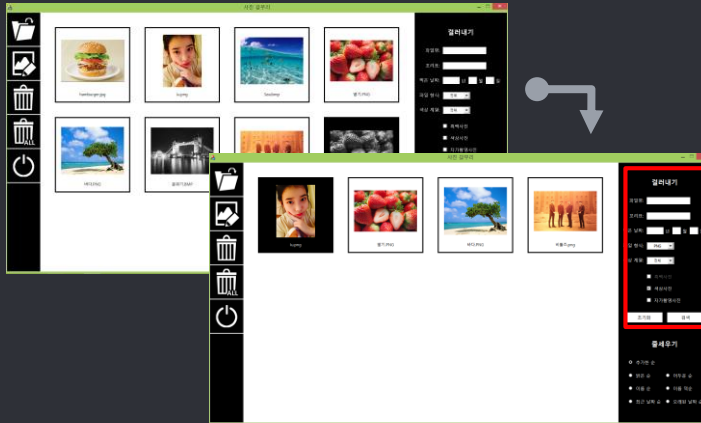
깔끔한 디자인을 구현하기 위해 MFC의 기본 폼을 커스터마이징하여 사용자 정의 뷰와 디자인을 구성했습니다.

### 후기

의도한 기능을 모두 구현하지 못한 것이 가장 아쉽습니다. 본래의 계획이었던 새로운 파일 포맷을 제대로 구현했다면 조금 더 재미있는 프로젝트가 됐을 것 같습니다. 또한, 이미지 특징 추출 부분에서 사람과 풍경, 실내와 실외, 자가촬영(셀카) 여부 등의 유용한 분류 기준을 적용하지 못하거나 인식률이 낮은 것이 아쉽습니다. 특히, 자가촬영 여부는 단순히 피부색에 해당하는 픽셀수가 전체 사진의 특정 비율 이상일 때로 판단을 했는데, 이는 얼굴 모양 패턴 인식을 이용했다면 좋은 인식률을 가질 수 있을 것입니다. 이 기술을 적용하지 못한 것이 아쉽습니다.

프로젝트의 난이도나 완성도는 높지 않지만, 걸핌기로 공부했던 MFC를 파헤칠 수 있는 좋은 기회였습니다. 재사용 할 수 있는 사용자 정의 뷰를 생각해보기도 하고, 어떻게 설계하면 MFC의 구조를 효율적으로 활용하고 객체지향적으로 적용할까를 고민해 보기도 하면서 재미있게 진행한 프로젝트였습니다.

### 5-3. '사진 관리 프로그램 갈무리' 실행화면



#### 컬러내기

파일명:

꼬리표:

찍은 날짜:  년  월  일

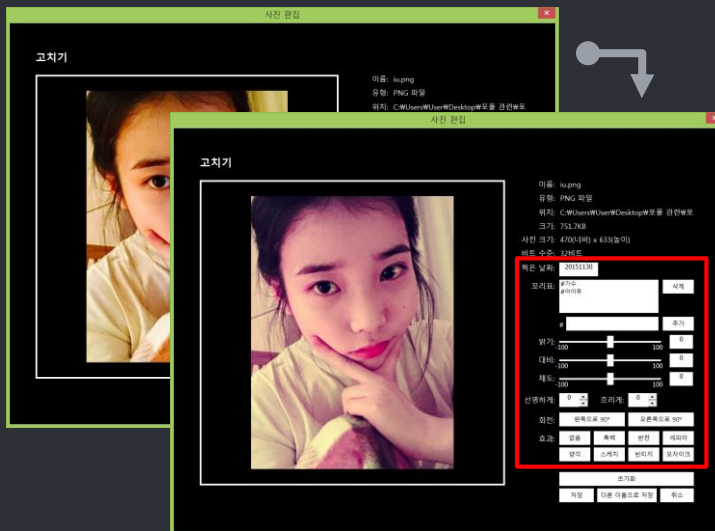
파일 형식: **PNG**

색상 계열: **전체**

☐ 흑백사진  
☒ **색상사진**  
☐ 자가촬영사진

컬러내기 작업을 통해 원하는 옵션들을 만족하는 사진만 검색해 정렬할 수 있습니다.

이 실행화면은 PNG 파일 형식, 컬러사진으로 옵션을 설정해 검색한 모습입니다.



찍은 날짜: 20151130

꼬리표: #가수 #아이유

#

밝기:  -100 100 0

대비:  -100 100 0

채도:  -100 100 0

선명하게: 0 흐리게: 0

회전:

효과:

고치기 작업을 통해 사진을 간단하게 편집하고, 찍은 날짜를 수정하거나 꼬리표를 삽입해 검색을 위한 옵션을 추가할 수 있습니다.

이 실행화면은 꼬리표를 추가하고, 세피아 필터 효과를 적용한 모습입니다.

## 6

# Z사 iOS 개발 테스트

- 6-1. 프로젝트 소개
- 6-2. 구현과정과 후기
- 6-3. 실행화면

※ GitHub 내 소스위치: 'university-portfolio' repository -> ZComTest



## 6-1. Z사 iOS 개발 테스트 프로젝트 소개

### 프로젝트 소개

이 프로젝트는 Z사 입사지원 테스트로 진행했던 iOS 개발 역량 테스트입니다. 주어진 요구사항과 우대사항에 따라 아이폰 앱을 제작했습니다.

구글 맵 기반 맵 뷰, 테이블 뷰, 네비게이션 컨트롤러를 이용해 전체적인 UI를 구성했습니다. 구글 맵 내 마커는 이미지로 구성되어 있으며, 줌값에 따라 이미지가 변경됩니다. 마커를 클릭하거나, 목록에서 항목을 선택해 상세정보를 확인할 수 있습니다. 또한, 목록은 원하는 옵션에 따라 정렬을 할 수 있습니다.

앱에 쓰이는 모든 데이터는 주어진 서버 URL로부터 JSON 형태로 받아 파싱해 사용했습니다. 이미지 캐싱 및 데이터 캐싱을 지원하도록 제작했습니다. 외부 오픈소스 라이브러리 관리는 CocoaPods를 이용했습니다.

### 개발기간 및 인원

#### 개발기간

2017.03.22  
~ 2017.03.26

개발인원: 1명

### 개발환경

#### 운영체제

macOS Sierra

#### 개발언어

Swift 3

#### 개발도구

Xcode 8.3

#### 기타

iOS 10.3 기반  
아이폰

## 6-2. Z사 iOS 개발 테스트 구현과정과 후기

### 구현과정

제가 최근에 Swift 프로그래밍을 주로 하기 때문에 이 테스트 앱은 온전히 Swift로 만들어졌습니다. UI는 깔끔하게 표현하기 위해 목적에 맞는 뷰를 선택했으며, **오토 레이아웃**을 사용해 만들었습니다. 또한, 유용하고 신뢰성 있는 오픈소스를 여럿 활용해 제작했습니다.

**GoogleMaps** 오픈소스는 구글 맵을 이용하기 위해 사용했습니다. 모든 이미지 처리는 **Kingfisher** 오픈소스를 이용해 비동기적 갱신을 하며, 이미지 캐싱을 지원합니다.

모든 데이터는 **Alamofire** 오픈소스를 이용해 URL로부터 JSON 형태로 받으며, 간단한 파싱을 위해 **SwiftyJSON** 오픈소스를 사용했습니다.

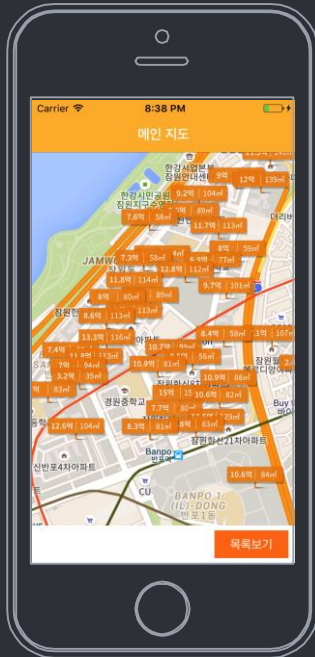
전송 받은 데이터는 가볍고 빠른 **RealmSwift** 오픈소스를 사용해 로컬 DB에 객체 매핑 방식으로 저장합니다. 관계형 DB와 다르게 객체 간 관계를 기반으로 모델을 객체처럼 관리할 수 있습니다. 모든 데이터는 로컬 DB를 통해 제공되며, 데이터가 존재하지 않는 경우에만 서버로부터 전송 받습니다.

### 후기

전체적으로 이미지를 불러오는 성능이 아쉬웠습니다. 처음 앱을 실행시켰을 때, 맵 뷰에 마커 이미지를 로딩하는 속도가 눈에 띄는 정도입니다. 목록의 이미지 또한 한 번 불러오고 나면 이미지 캐싱이 되기 때문에 빠르지만, 처음 로딩의 속도가 느린 것이 눈에 띄었습니다. 맵 뷰 내 마커의 경우, 기획상 처음부터 맵 뷰를 띄우기 때문에 어쩔 수 없지만, 기획이 변경된다면 미리 이미지를 로딩하는 식으로 해결할 수 있을 것 같습니다. 목록 내 이미지의 경우, 처음부터 파일 크기가 작은 이미지를 준비하거나 기획에 따라 미리 로딩하는 방식을 택해야 할 것 같습니다.

많은 오픈소스를 접하고 사용해볼 수 있는 좋은 기회였습니다. 바퀴를 다시 발명하지 마라는 말처럼 신뢰성 있는 오픈소스가 있다면, 적극 활용하는 것이 좋다고 생각합니다. 꾸준히 다양한 오픈소스에 대해 알아보는 것이 중요하다는 생각이 들었습니다.

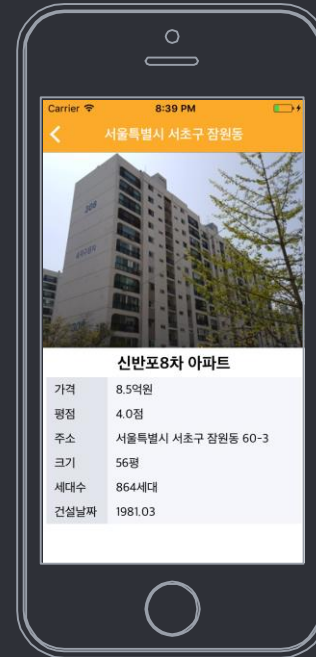
## 6-3. Z사 iOS 개발 테스트 실행화면



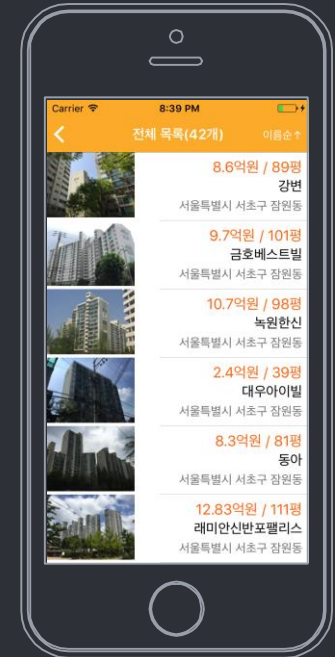
시작 화면



줌 값에 따른  
마커 변화



상세정보



전체 목록



읽어주셔서  
감사합니다 !