

공개 소프트웨어 프로젝트 최종보고서

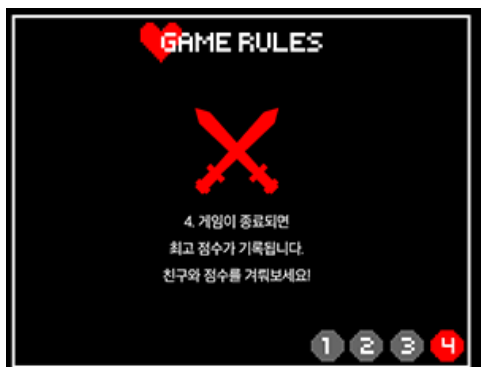
과제 수행원 현황						
수행 학기	<input type="checkbox"/> 2019년 1월~6월 <input checked="" type="checkbox"/> 2019년 3월~6월 <input type="checkbox"/> 2019년 9월~2019년 12월					
프로젝트명	Face tracker을 이용한 얼굴 인식 게임 YAM-YAM					
팀명	openthedoor					
	학과	학번	성명	성별	연락처	E-mail
팀장	컴퓨터공학과	2017112106	신소희	여	010 9613 3466	skysohe@naver.com
팀원	컴퓨터공학과	2017112065	김예지	여	010 9945 2807	dpwl2807@naver.com
	컴퓨터공학과	2017112082	김혜지	여	010 2676 4338	sosox00@naver.com
	컴퓨터공학과	2017112076	이미란	여	010 8784 7803	rannan09@naver.com
지도교수	교과목명	공개SW프로젝트				
	소속	<input checked="" type="checkbox"/> 컴퓨터공학전공 <input type="checkbox"/> 정보통신공학전공 <input type="checkbox"/> 멀티미디어공학전공 <input type="checkbox"/> 융합소프트웨어연계전공				
	성명	손윤식 교수님				

# Key Words	Face tracker	OpenCV	OpenGL	Node.js	AWS(EC2, RDS)
1.개발동기/목적/필요성 및 개발 목표	<p><개발동기 및 목적></p> <p>오픈소스를 이용하여 프로젝트를 진행하는 것이 목적이었기 때문에 여러 오픈소스를 찾아보다가 OpenCV를 이용한 얼굴 인식 Face Tracker 오픈소스를 발견하게 되었다. 처음 이 오픈 소스를 봤을 때 가진 생각은 정말 재밌는 프로젝트를 만들 수 있겠구나였다. 따라서 이 오픈 소스를 이용하여 어떤 프로젝트를 진행할 수 있을지 생각해보았다. 그러다가 어릴 때 사용하던 2g폰에 있던 과자 먹기 게임을 재밌게 했던 기억이 났고 이 Face Tracker로 그러한 추억을 회상할 수 있도록 하고, 레트로 감성을 살려서 우리만의 eating game을 만들 수 있겠다는 생각을 하였다. 따라서 우리 팀은 Face Tracker 오픈소스를 가지고 여러 도형이 떨어지고 그것을 입으로 먹을 수 있는 게임을 개발하기로 하였다. 또한 게임에</p>				

2.최종 결과물 소개	<p>어떤 요소를 추가할 수 있을지 고민하다 게임하면 경쟁이 빠질 수 없기에 친구들과 최고기록을 가지고 경쟁할 수 있게끔 하기 위해 서버와 자체 DB를 구현하기로 하였다.</p> <p><개발목표></p> <p>이 프로젝트를 구현함으로써 사용자에게 예전의 경험을 회상하게 하고, 그 기억을 떠올리며 예전의 게임과 비슷하지만 새로운 게임을 경험해 볼 수 있게 한다. 또한 게임 시작 전에 게임의 규칙을 알려주어 게임 진행법을 어렵지 않게 숙지할 수 있게 한다. 게임 중간중간 떨어지는 장애물들을 추가하고 피버타임을 주어 게임에 대한 흥미를 높인다. 회원가입과 로그인 절차를 두고 랭킹을 보여주는 기능을 통해 본인의 기록과 다른 사용자와의 점수 차이를 볼 수 있고 승부욕을 자극할 수도 있다.</p>
	<p><최종결과물 소개 및 사진></p> <div data-bbox="392 880 999 1223" data-label="Image"> </div> <p>- 처음 프로그램을 시작하면 [1]회원가입과 [2]로그인을 할 수 있는 화면이 보인다. 위 사진은 회원 가입하는 화면이다. 비밀번호가 암호화되어 *표시로 출력되는 것을 확인할 수 있다. 이 암호화된 비밀번호는 해싱되어 DB에 들어간다.</p> <div data-bbox="392 1518 999 1861" data-label="Image"> </div> <p>- 회원가입이 완료되면 그 아이디 비밀번호로 로그인이 가능하다. 여기서도 마찬가지로 비밀번호가 *표시로 출력됨을 확인할 수 있다.</p>



- 로그인 완료되면 게임의 첫 화면으로 넘어간다. 이 첫 화면에는 게임의 제목과 START버튼, gamerules버튼, RANKING버튼, EXIT버튼이 있다.



- gamerules버튼을 클릭하게 되면 위 사진들과 같이 단계별로 게임의 규칙들이 설명이 되고 아래 번호들을 클릭하면서 해당 순서에 맞는 규칙을 확인할 수 있다.



- 게임 첫 화면에서 start버튼을 클릭하면 이렇게 게임이 시작된다. 게임이 시작되면 배경음악이 함께 플레이 된다. 도형이 입에 정확하게 들어가게 되면 이팅사운드와 함께 해당도형에 설정된 설정 값이 적용된다. 사진을 보면 오른쪽 위에는 스코어가 출력되고 오른쪽 아래에는 남아있는 생명이 나타나 있는 것을 확인할 수 있다. 또한 특정 점수 이상이 되면 맨 마지막 사진처럼 BURNING이라는 표시와 함께 장애물들은 모두 사라지고 득점 가능한 사각형과 원만 떨어지는 피버타임이 주어진다. 이렇게 게임을 진행하다가 생명이 모두 없어지면 게임은 종료된다.

게임 진행중에 얼굴이 빨강게 표시한 사각형 영역을 벗어나게 되면 피버타임때와 같이 사각형 내부가 붉게 표시되면서 경고를 표시하고 얼굴을 재인식한다. 이러한 기능은 게임 진행중에 얼굴 인식이 잘 되지 않아 게임이 원활하게 진행되지 않을 상황을 방지하기 위함이다.



- 게임이 종료되면 방금 진행한 게임에서의 점수를 확인할 수 있고 지금까지의 본인의 최고점을 확인할 수 있다.



- 여기서 RANKING버튼을 누르게 되면 이 게임에 가입 되어있는 다른 사용들과 본인의 랭킹을 확인할 수 있다.

<p>3.프로젝트 추진 내용</p>	<p><프로젝트 진행과정></p> <p>우리의 프로젝트는 Face tracker라는 오픈소스를 이용하여 도형 먹기 게임을 만드는 것이다. 프로젝트는 크게 두개의 파트로 구분되어 진행되었다.</p> <p>핵심 오픈소스인 Face tracker를 이용한 메인 게임 코드 구현, Node.js를 이용한 서버 구현이다.</p> <p>첫째로 Face tracker를 이용한 메인 게임 코드 구현의 진행과정은 다음과 같다.</p> <ol style="list-style-type: none"> 1. Face tracker 설치 및 실행 2. Face tracker 코드 & 파일 구조 분석 3. Face tracker를 사용하여 입과 입의 움직임을 인식하게 하는 방법 구상 4. Open GL을 사용하여 영상위에 랜덤하게 떨어지는 도형을 구현하기 5. 입과 도형이 닿았을 때 발생할 이벤트 구상 6. Face tracker가 정상적으로 작동하지 않는 상황에 대한 이벤트 처리 <p>우리의 프로젝트가 오픈소스를 이용한 "게임" 만들기이기 때문에 게임에 필수적으로 필요한 요소를 만족시키기 위한 추가적인 설계 진행 과정은 다음과 같다.</p> <ol style="list-style-type: none"> 1. 랜덤하게 떨어지는 4가지 성격의 도형이 입에 닿을 때마다의 점수 처리, 게임이 종료되기 위한 생명 아이템 체제 도입 2. 유저의 긴박감을 유발하기 위한 추가 게임 요소 구현 3. 유저의 경쟁 심리를 자극하기 위한 랭크 체제 도입 4. 게임 UI구현 및 사운드 추가 <p>다음으로는 서버 구현을 위한 진행 과정이다.</p> <ol style="list-style-type: none"> 1. Node.js 설치 및 작동 확인 2. Node.js, MySQL 이해 및 예제 실행 3. AWS EC2, RDS 인스턴스 생성 및 연동 4. 서버 동작 코드 작성, 테스트 후 서버에 올리기 5. 메인 게임 소스와 연결하기위한 socket 코드 작성 6. 로그인, 회원가입 시 유저 정보 예외 처리 7. 비밀번호 뷰 암호화 8. Face tracker과 연결 후 최종 실행 파일 생성
--------------------------------	---

<프로젝트 구현과정>

프로젝트의 구현과정은 위에서 설명한 진행과정의 절차에 따라 기술하겠다.

1. Face tracker 설치 및 실행

사용한 오픈소스의 작동 환경이 Ubuntu였기 때문에 팀원들 모두 Ubuntu를 설치하였다. Open CV를 리눅스에 설치하는 패키지 명령어는 16.04 버전까지 지원했기 때문에 기존에 18.04 버전을 사용하던 팀원을 포함하여 모두 16.04로 버전을 통일하였다. 그 후 Face tracker를 설치한 후 실행하였다.

make 명령어를 통해 생성된 Face tracker 실행 파일이 시작과 동시에 종료되는 문제가 발생하였다.

문제 발생 원인과 해결 방안은 다음과 같다.

1) 코드 오류: std 내 스트링을 지원하지 않음 -> cc파일에서 cstring.h 파일을 include

2) 카메라 오류:

카메라 작동을 확인하기 위해 Ubuntu의 카메라 패키지인 cheese를 실행해보았으나 해당 프로그램에서도 카메라가 정상적으로 작동되지 않았기에 우분투에서 노트북의 내장카메라가 인식되지 못함을 확인하였다.

카메라 인식 문제: virtual box의 기본 버전이 웹 캠을 인식하지 못함을 발견 -> extension pack을 추가 설치하여 vBoxManage를 통해 webcam을 수동으로 Attach 후, 카메라 작동을 성공시켰다. 또한 가상머신을 사용하여 발생하는 버퍼링과 싱크 문제를 해결하기 위하여 외부 장치에 운영체제를 우분투로 설치하여 문제를 해결하였다.

2. Face tracker 코드 & 파일 구조 분석

Face tracker를 사용하기위해 첫번째로 코드와 파일의 구조를 분석하였다.

- 코드분석(face_tracker.cc)

-> main문 포함

-> void draw() : 화면에 출력될 선, 점 구현 / triangulation을 이용해서 거리 측정하여 위치측정

-> int parse_cmd() : facetracker실행시 옵션을 줬다면 해당하는 명령 수행

-> main() : 카메라를 실행시키고 화면에 출력

-> 헤더파일 : Tracker.h (face tracker와 관련된 모든 함수 -> lib에 구현)

OpenCV/highgui.h (OpenCV가 지원하는 카메라 관련 함수)

- 파일구조

Face Tracker

-> bin : 실행파일

-> include : 헤더파일(.h)

-> src : exe(face_tracker.cc), lib(여러 .cc 파일)

FACE TRACKER

파일구조

FaceTracker

- ↳ bin - 실행파일
- ↳ include - 헤더파일(.h)
- ↳ src
 - ↳ exe
 - ↳ face_tracker.cc
 - ↳ lib
 - 여러.cc 파일

face_tracker.cc

: main문 포함

void draw

: 화면에 출력될 선, 점, 구형 등
Tranquilization을 이용해 처리해 줌

int parse_cmd

: 페이스트레이커 실행시 옵션을 줘야
해당하는 명령 수행

```

$ cd /home/ranlee/VirtualBox
$ ./face_tracker --help
face_tracker: written by Jason Saraghi 2018
Performs automatic face tracking

usage: ./face_tracker [options]

Arguments:
  -s <string> -> Tracker model (default: ../model/face2.tracker)
  -c <string> -> Connectivity (default: ../model/face.con)
  -t <string> -> Transformation (default: ../model/face.tri)
  -s <double> -> Image scaling (default: 1)
  -d <int> -> Frames/detections (default: -1)
  -check -> Check for failure
    
```

→ 경로를 따로 지정하는 것 같았다. 쓸일 없을 듯

main

: 우리는 여기에 코드 주어야 함!

: 카메라를 실행시키고 화면에 출력

헤더파일

Tracker.h : 페이스트레이커와 관련된 모든 함수

opencv/highgui.h : opencv가 지원하는 카메라 관련 함수

include

공통으로 있는 함수 → Init, Load, Save, Write, Read

Fcheck.h

→ 트레이킹 실패 체크

Fdet.h

→ OpenCV의 face detector를 강방
이용해서 Face_tracker가 사용할 수 있게

IO.h

→ Input/Output operation

PAW.h

→ 조각처럼 만듦 → main에서 사용

PDM.h

→ 3D 분산 모델

CLM.h

→ 저명한 Local Model

Tracker.h

→ CLM.h, FDet.h, Fcheck.h

Patch, FDet, PAW, PDM,

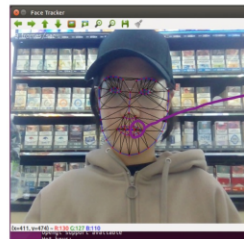
IO.h

→ CLM

→ PDM.h, Patch.h

Fcheck

→ PAW.h



→ main의 draw
함수에서 check로
분리된 부분, 빨간
점들을 받아와서
근기변환을 적용하
도록 구현해야
한 것 같다.

그림 1 Face Tracker 코드 & 파일 구조 분석

3. Face tracker를 사용하여 입과 입의 움직임 인식하게 하는 방법 구상

눈, 코, 입, 얼굴형 전부를 인식하는 Face tracker에서 우리는 입 영역만 필요하였다. 부가적인 요소로 얼굴 색상 변경이 있지만 핵심 부분인 입 영역 먼저 추출하는 것이 먼저였다.

-> face tracker의 구성 요소 중 빨간 동그라미 부분이 계속해서 움직이는 영역을 포착함을 확인하였고 이를 활용하면 원하는 영역만 추출할 수 있을 것이라고 생각하였다. 따라서 circle마다 인덱싱을 추가하였다.

-> 입 안 영역을 인식하는 circle의 인덱스는 60, 61, 62, 63, 64, 65임을 확인하였고 여러 번의 실행을 통해 이는 계속해서 바뀌지 않는 고유의 인덱스임을 확인하였다. 이로써 입 영역 구분 성공하였다.

-> 입 영역이 제대로 인식되고 있는지 확인하기 위하여 해당 영역을 빨간 다각형으로 시각화하였다.

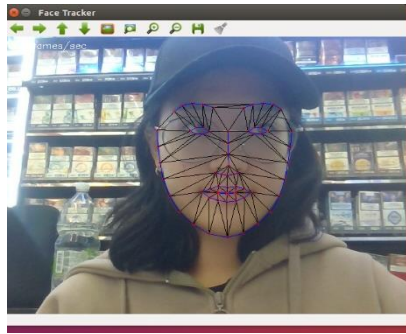


그림 2 기존의 Face Tracker

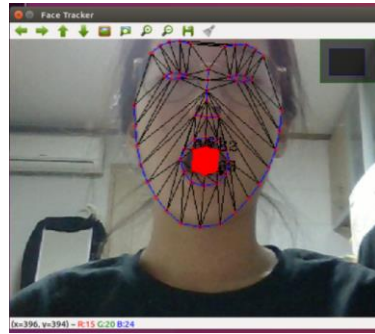


그림 3 입영역 구분을 추가한 Face Tracker

4. Open GL을 사용하여 영상위에 랜덤하게 떨어지는 도형을 구현

첫째로 OpenGL을 Ubuntu에 설치하고 정상적으로 실행시키는 것이 필요했다. 기존 Face tracker는 makeFile을 통해 컴파일 되는데 OpenGL 또한 컴파일 시에 옵션이 필요한 상태였다.

->FaceTracker의 makeFile은 아주 복잡한 구조로 이루어졌기 때문에 우선 MakeFile 분석을 통해 FaceTracker가 어떻게 컴파일 되는지 분석하고, OpenGL의 라이브러리를 포함시킬 위치를 찾아 OpenGL 컴파일을 성공시켰다.

GLUT라는 OpenGL용 유틸리티 툴킷을 사용하려 했으나 이번에 시도해본 원 그리기나 위에서 점 떨어지기 수준은 OpenGL 라이브러리를 이용해서 구현하였고, 점점 더 어려운 애니메이션 구현을 위해 기본적인 점, 선, 면 표현을 지원하고 프로그램 창의 생성과 설정을 지원하는 GLUT도 공부하기로 하였다.

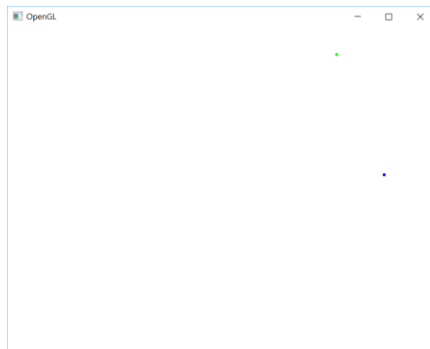


그림 4 OpenGL을 이용해 점 떨어뜨리기


 원그리기



그림 5 OpenGL로 원 그리기

게임 진행을 위해서는 도형이 “랜덤하게” 떨어져야 했으므로 OpenGL을 이용하여 랜덤하게 떨어지는 물체를 구현하였다.

main 함수 시작 → glutMainLoop 실행 → timerFunction → RenderScene 순으로 진행시켰다.



그림 6 랜덤하게 떨어지는 물체 구현

랜덤하게 떨어지는 물체를 구현하는 과정에서 문제점이 발생했다.

랜덤 한 위치(좌, 우, 위)에서 랜덤 한 속도와 랜덤 한 방향으로 떨어지게 구현했지만, 세 점 모두 x방향의 위치만 다를 뿐 같은 속도와 같은 패턴으로만 떨어지게 된다는 것이었다. 따라서 이 문제를 아래와 같이 해결하였다.

실행할 때마다 항상 왼쪽에서 출력되는 것의 문제가 그 위치를 랜덤하게 설정해주는 변수 a를 전역변수로 실행하여 실행할 때 초기화가 되지 않기 때문이라는 것을 발견하여 aSelect()라는 함수에서 따로 a를 선언하여 초기화 해주었더니 도형들이 랜덤 한 위치에서 출력되었다.

도형을 먹으려고 할 때 도형이 너무 랜덤하게 움직여서 아예 입 근처로 오지 않고 사라지는 경우가 있을 것이라고 예상했다. 그런 도형들을 먹으려고 얼굴을 너무 크게 움직이면 얼굴인식이 잘 안되는 경우가 있을 것이라고 예상했기 때문에 이 문제를 해결하기 위해 왼쪽, 오른쪽에서 날아오는 도형은 위, 아래로 절반 구역을 나눠서 위쪽에서 나오는 것은 첫 움직임이 무조건 아래쪽으로 향하게, 아래에서 나오는 도형은 첫 움직임이 위로 향하도록 설정하였다. 또 위에서 떨어지는 도형들도 왼쪽, 오른쪽으로 절반씩 구역을 나눠서 왼쪽에서 떨어지는 도형은 첫 움직임이 오른쪽으로 가도록, 오른쪽에서 떨어지는 것은 왼쪽으로 가도록 설정하였다. 도형을 먹기가 좀 더 쉽게 특정 범위를 설정해서 그 범위에서는 직선으로만 움직이게 해서 전에 보다는 도형이 먹기가 쉽게 되었다.

우리는 이제 떨어지는 도형을 기존의 FaceTracker와 합치는 것만이 남았지만, 이는 쉽지 않은 여정이었고 프로젝트 내에서 가장 어려웠던 문제점이라고 생각한다. 이를 어떻게 해결하였는지 설명한다.

OpenGL로 구현한 도형을 OpenCV에서 이용하기 위해서는 OpenCV 영상을 받아와 OpenGL로 도형을 그리고 다시 OpenCV 윈도우에서 출력하는 과정이 필요하였다. 이를 위해 OpenGL의 편리한 활용을 위해 만들어진 GLUT 툴 사

용하였다.

→ Display()함수 작성하고, 임시로 움직이는 도형 생성한다.

➔ OpenGL로 배경을 검게 하여 움직이는 입체 도형을 그리고 도형만 따낸 다음 마스크를 지정하여 OpenCV 이미지에 겹쳐 그리는 아이디어를 생각해 적용하였다.

→ 따라서 OpenCV 변수로 웹 캠의 영상을 받아오는 Mat변수를 OpenCV의 Ipl변수로 변환하고, 그 후 glReadPixels 함수를 이용하여 OpenGL로 그린 그림이 Ipl이미지에 출력될 수 있게 하였다. OpenCV의 윈도우에 출력을 위해 다시 Mat 변수로 변환한다.

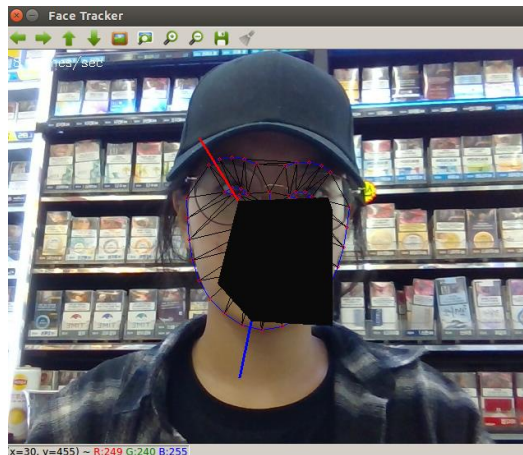


그림 7 임시로 회전하는 정육면체를 Face Tracker에 출력

위와 같은 방식으로 구현하면 될 줄 알았지만 구현을 마치고 보니 OpenGL의 애니메이션 기능을 사용하기 위해선 glutTimerFunc()함수가 필수불가결하였다. 하지만 이 함수는 glutmainloop()와 맞돌아 작동하는데 glutmainloop()함수를 처음에 사용하지 않은 이유는 해당 함수를 실행하면 return없이 계속 OpenGL이 작동하기 때문에 계속해서 변하는 웹카메라의 FaceTracker를 받아오지 못해서였다. 하지만 날아오는 물체를 구현하기 위해서 glutmainloop()함수를 무조건 이용해야 하므로 OpenCV의 기능을 전면 OpenGL로 이동하였다. 따라서 OpenCV의 영상을 OpenGL형 변수로 변환하여 해당 영상을 background로 설정하는 방법을 생각하였다. 따라서 형변환을 위한 MatToTexture 함수를 작성하였고 텍스처 바인딩을 통해 OpenGL의 배경으로 포함시켰다. 또한 기존 main의 while(1)에 있던 코드들을 옮기는데 많은 고민이 필요했는데 timer함수를 1초마다 불러들이게 해 while문과 같이 작동하도록 하여 timer 함수로 main 무한루프에 있던 코드를 옮겼다. 그리고 기존 FaceTracker 코드는 키보드의 esc버튼만을 통해 종료할 수 있고, 키보드의 d를 입력 받아 frameReset이 가능하였다. 하지만 GlutMainLoop()가 실행되면 반환형 없이 OpenGL 코드들만 실행되기 때문에 해당 코드 이후에 작성하는 것은 의미가 없었다. 하지만 glutKeyboardFunc를 이용하여 keyboard의 signal이 오면 해당 입력 값을 받아 실행시키도록 하여 기존 FaceTracker와 완전히 동일한 기능을 수행하도

록 OpenGL 코드로의 변환에 성공하였다.

그 후 발생한 문제점은 OpenCV로 작성된 Face tracker를 OpenGL 코드로 변경하여 실행했을 때 좌표계가 맞지 않는다는 문제였다.



그림 8

그림 8은 맨 위 왼쪽의 검은색 점처럼 보이는 것이 OpenCV 화면, 그 보다 아래에 있는 검은색 동그라미가 OpenGL화면이다. 화면이 맞지 않는 이유가 OpenCV와 OpenGL의 좌표계가 다르기 때문이라는 결론을 내렸다.

→ 원의 크기와 gluOrtho2D라는 함수를 조정하여 어느정도 조정은 했으나 두 환경의 좌표계에 대해 계산을 하여 CVtoGL 함수를 구현해 OpenGL에 맞는 좌표 계 계산하는 함수를 구현하여 더 적합한 화면이 나올 수 있도록 조정을 하였다.

위의 문제를 해결한 후 OpenGL을 좀 더 공부하였다. 랜덤 한 수를 생성하여 상하좌우, 랜덤 한 곳에서 색상을 변경하며 도형이 날아오게 하였다. 좌표 값을 계산하여 도형이 상, 하, 좌, 우 중 어느 한곳과 부딪히면 사라지게 하였다. 계속해서 Timer 함수를 호출하여 끊임없이 나오도록 하여 종료하지 않도록 하였다. 해당 함수와 Face tracker를 결합하여 Face tracker 화면에 랜덤으로 날아오는 도형을 출력할 수 있었다.



그림 9 최종적으로 떨어지는 도형과 Face Tracker를 합친 영상

5. 입과 도형이 닿았을 때 발생할 이벤트 구상

FaceTracker의 draw함수에서 입 부분의 인덱스를 배열에 저장한다. 그리고 계속해서 redirect되는 display함수에 도형의 x값과 y값이 위에서 얻은 배열 내에 위치하는지 판단한다. 만약 도형의 중심이 해당 배열 내에 있다면 반지름을 0으로 바꿔 도형이 사라지게 한다. 다른 도형을 추가해서 구현할 때 중심의 좌표를 따질 수 없다면, 모서리의 좌표를 따지거나 아니면 아예 도형의 위치를 화면 밖으로 보내는 방법을 생각하였다.

반지름을 0으로 바꾸고, 전역변수로 설정한 score 변수도 증가시켜 점수가 올라가게 하였고, SCORE와 LIFE를 화면에 출력하여 현재 스코어와 생명이 몇 개 남았는지 확인 가능하게 하였다. 또한 도형마다 이벤트 처리를 해주었다. 각 도형마다 고유한 중심 좌표 값이 있거나 꼭지점 좌표 값이 있기 때문에 중심 값인 경우는 해당 좌표가 입 영역에 들어올 때 처리를 해주고, 꼭지점 좌표 값인 경우는 중심으로부터의 거리를 계산하여 입에 닿을 때 이벤트를 처리해주었다.

6. Face tracker가 정상적으로 작동하지 않는 상황에 대한 이벤트 처리

페이스 트레이커는 얼굴이 화면밖을 나가면 재인식하는 과정이 필요하다. 이와 같은 과정은 게임진행에 방해가 된다고 생각했기 때문에 지정된 영역 밖으로 얼굴이 나가게 되면 경고를 표시하기 위해 위와 같은 사진처럼 빨간 영역을 설정하였다. 얼굴이 영역을 밖을 나가면 불투명한 빨간 상자가 나타난다. 평상시에는 빨간 선으로 영역 표시만 되어있다. 이와 같은 영역은 OpenCV의 관심영역 지정을 통해 알파 값을 조정하여 구현하였다. 또한 해당 영역 밖을 나간 경우 프레임 재인식을 통해 Face Tracker로 얼굴을 다시 인식하도록하였다.



그림 10 페이스 가이드 라인 설치

우리의 프로젝트가 오픈소스를 이용한 “게임” 만들기이기 때문에 게임에 필수적으로 필요한 요소를 만족시키기 위한 추가적인 설계 진행 과정은 다음과 같다.

1. 랜덤하게 떨어지는 4가지 성격의 도형이 입에 닿을 때마다의 점수 처리, 게임이 종료되기 위한 생명 아이템 체제 도입

기존의 원, 사각형에 추가하여 삼각형과 역삼각형을 구현하였고 Face tracker 창에 출력시켰다. 게임의 시나리오를 바꿔서 원과 사각형을 먹으면 점수가 올라가고 삼각형을 먹으면 생명을 주고 (최대 3개) 역삼각형을 먹으면 생명이 감소되도록 구현하였다. 또한 네 개의 도형의 색이 다르게 출력되게 하기위해 각 도형별로 첫 colorIndex를 다르게 설정하였다.

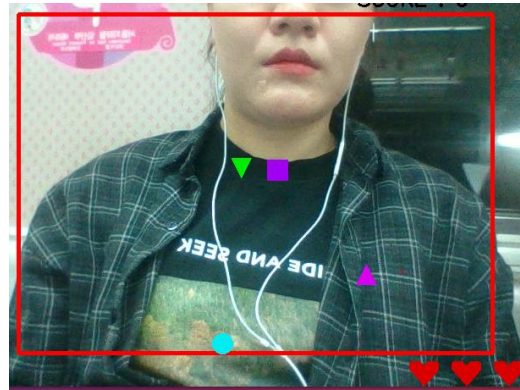


그림 11 생명 아이템 구현

2. 유저의 긴박감을 유발하기 위한 추가 게임 요소 구현

점수에 따라 도형의 속도를 조절하여 게임이 진행될수록 점점 어렵게 조절하였다. 생명을 증가시키는 삼각형은 처음부터 특히 더 속도를 빠르게 했다.

게임적인 요소를 더 추가하기 위해 특정 점수에 도달할 때마다 보너스 타임으로 원만 많이 떨어지도록 구현하였다. 원을 여러 개 생성해야 하기 때문에 원을 그리고 그 원을 아래로만 떨어지게 하는 클래스를 구현하였고 이 원은 먹으면 10점씩 올라가도록 설계하였다.

3. 유저의 경쟁 심리를 자극하기 위한 랭크 체제 도입

유저가 게임을 끝내면 얻은 점수와 지금까지 모든 유저 중 최고점을 보여주는 것에만 그치지 않고 랭킹을 매겨 직접 확인할 수 있게 설계하였다. 게임 시작 화면의 RANKING 버튼을 클릭하면 게임이 시작하기 전에 먼저 랭킹을 확인 가능하며 게임이 끝난 후에도 확인할 수 있도록 설계하였다.

4. 게임 UI구현 및 사운드 추가

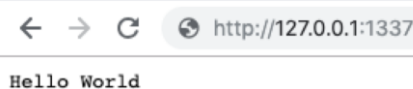
게임을 좌우하는 하나의 요소는 미적 요소이기 때문에 사용자가 보기 좋고 매력을 느낄 만한 화면을 제작하였다. 포토샵을 사용 & OpenCV를 이용하여 마우스 콜백 함수를 작성해 마우스 클릭 시 이벤트에 따라 윈도우에 출력되는 화면이 달라지도록 설계하였다.

다음으로는 서버 구현을 위한 진행 과정이다.

1. Node.js 설치 및 작동 확인

2. Node.js, MySQL 이해 및 예제 실행

팀원들 모두 Node.js에 대한 이해도가 없었으므로 일단 유튜브나 생활코딩같은 플랫폼을 통해 강좌를 찾아 공부하는 시간이 필요했다. 영상을 보며 route와 정적 파일을 다루는 법, template으로 jade와 ejs공부, get/post, 파일을 통해 웹 애플리케이션 개발을 공부하고 시도해보았다 배운 내용에 대해 간단한 예제를 통해 확인해보았다.

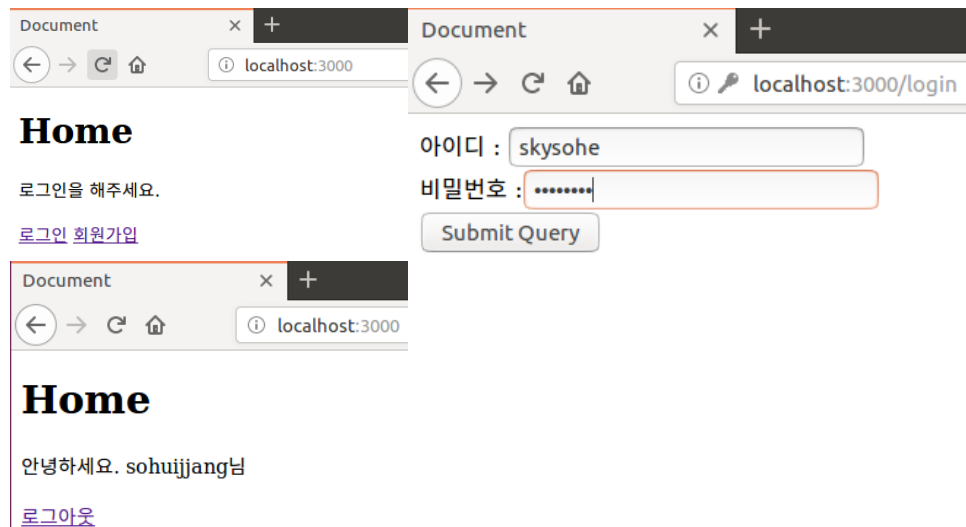


test.js 파일을 작성하여 createServer 함수로 서버를 만들어 포트에 연결시켜 "Hello World"를 출력해보는 아주 간단한 예제부터 공부를 시작하였다.

<Node.js 예제 1 실행>

서버 파트의 팀원 간 Node.js 버전을 v8.9.4로 통일하여 설치하였다.

서버 파트에서 가장 확실하게 구현해야 할 부분은 회원가입과 로그인 그리고 점수를 이용하여 랭킹 매기기였다. 웹 view를 이용하여 정확하게 작동하는지 테스트하며 기능을 구현해 나갔다.



<간단하게 로그인 기능을 구현 해본 결과>

서버와 데이터베이스 간 사용될 언어는 MySQL이었기 때문에 MySQL에 대한 공부 또한 틈틈이 하였다. 다행히 MySQL은 지난 학기에 웹프로그래밍을 수강하며 공부한 적이 있었기 때문에 어렵지 않게 다룰 수 있었다. 이것 또한 Node.js에서 MySQL로 쿼리를 날려보며 작동 방법을 확인하였다.


```
Database changed
mysql> show tables;
+-----+
| Tables_in_testdb |
+-----+
| sohui             |
| test_db           |
| user              |
+-----+
3 rows in set (0.00 sec)

mysql> select * from user;
+-----+-----+
| userID | userPW |
+-----+-----+
| abc1233 | 123abc3 |
+-----+-----+
1 row in set (0.00 sec)
```

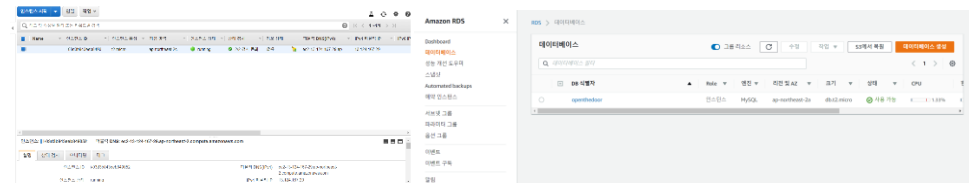
```
var connection = mysql.createConnection({
  host : 'localhost',
  port : 3306,
  user : 'root',
  password : '960113',
  database : 'testdb'
});
connection.connect(function (err) {
  if(err) console.log(err);
});
```

```
sohui@sohui-VirtualBox:~/open3$ node app.js
The solution is: [ RowDataPacket { userID: 'abc1233', userPW: '123abc3' } ]
```

<Node.js와 MySQL 간 작동 확인 결과>

3. AWS EC2, RDS 인스턴스 생성 및 연동

AWS의 EC2와 RDS의 무료 서버와 데이터베이스를 사용하기로 결정하였고, EC2 인스턴스를 생성하고 RDS의 엔드 포인트를 Workbench에 연결하였다.



<Openthedoor EC2, RDS>

4. 서버 동작 코드 작성, 테스트 후 서버에 올리기

Node.js에 대해 공부가 진행된 후 본격적으로 서버 작동 코드를 짜기 시작하였다. Workbench를 통해 스키마와 테이블을 생성하여 Postman으로 유저 정보를 넘겼을 때 정확히 작동됨을 확인한 후 서버에 코드를 하니씩 올리기 시작했다.

user_id	user_pw	user_salt	user_idx	score
yeji	OTt9Ur7vdOvMDLu4Gle8THagWkiFTHR1ZOMN...	IPjm482eKobrVxqKarc+ZUYIGIsW15FNCbdcRA84eY=	40	40
osp7	8cr8Ao/YK1E3067G01Bcv8zu3xGYRdE2ZJialTs...	SGeh5cxbriyQEFF65TjBUOaCdrszdJ8pxoxyDOyE98=	41	40
miranlee	oyhnpIgrNmbUk/ujBZV2Ta+W5SKomnz210RHX...	IFB5gCd0gOXUtwunVG23bSB13ISkEQEU+aPS24gX/dw=	43	310
hyex	WCcAR049vrJzBj4dUvk+HYg6BJtVwA/eQP0PH...	AdzcwYaQscX8E8cM3S8hQOE0R4wvvyZF66q2vfmMLI=	44	390
miran	qOly8HWeUFZ3tv/1h+Qhf9R8b5f8A1WHZVDJ...	M5BnUgl1Su1kUPwJruEknhlMGjUU+G3nINLAQUyL8=	50	40
sun	1ny6CQOfvZIVJAOnVGrARRCyzbAmUP0udHs...	X59Xp/pvBfW6rxj8JVoh9I7xMjUgneT2fa05kymo4vo=	52	40
sonbug	tmltzjghCp0y8ChI9P0ufJ6YDrBafGI1jWabYf3A...	+zskMMttCr867Vv8Yd+XxYIjWDCSCMT54AWx7KwFnY=	55	200
sjn	BBNSTrAYdtB1DmcOsuTIPT2uSLTtRuwf0URrFkg...	hF8KKuLIWACPBZeQwd6hxoA4mImpe3R4ax/tPwfs7s=	56	40
ran	gmwf8JIraWpxehGqCmSKIAiUBUHE+tOaRbYLx...	kabOqj80fajU9m7Wi+VbYlZuNW9mGtUgcFO8tJ63h8=	57	0
rann	2MfyA8c995SCIWko2aYnrgBCVoIQp0bAu4qKrX...	6Na1+RKeSMVLTYPG/ITY3yqfqrq6xbaPPx1JtLAUJ4=	58	0
miranl	Whgu4djdCxsCLvJMV7ppdhB6HvC5RZg7G3zkY...	NoJdiMbemI5kgA/SvPgeZctdeVvCBwPJYFV0fsi3yq8=	59	0
miran1	axXVZ91C5xQOV2ftOX9dgJhzItHDMwMLZZvXH...	GbnT+mFOCbLaCRw5pv/8jyESyAs/7G13hi482zr8/s=	60	0
mirann	mTDLpOXnrd/aICK14/PLmVvmPvsXTIIX+VjB4+	QYbgllhwuS23m+04eox43NGBOqCiOxbn9JYuvIKIUAQ=	61	0
miranle	0ZnHkZkmiUgg8/FfzkH7Ap5ubXTncEtDzhZItv/...	+YXED3cSymQg9HJq3L7QV4V54e8KC8pnVamrwivTTQ=	62	0
miran2	jmxlW/eLlMaGhsFivZABQZWZIdDohyIOzV06SV...	xdxYnWDYHhToqsQ2KuWd8cqHdbPVXBvYDR/KO6LU7ds=	63	0

<Postman과 Workbench를 통해 유저 정보가 정확히 들어옴을 확인>

```
app.js  config  node_modules  package.json  public  views
bin      module  npm-debug.log  package-lock.json
```



```

routes
├── first.js
├── index.js
├── ranking.js
├── rank.js
├── score.js
├── signin.js
├── signup.js
├── users.js
└── views
    ├── error.jade
    ├── index.jade
    └── layout.jade

```

<Openthedoor 서버의 디렉토리 구조>

완성된 서버 코드를 서버에 올리고 Node.js 모듈인 pm2로 구동 시켰다.

```
[ubuntu@ip-172-31-33-238:~/server/server$ pm2 list
```

Name	id	mode	status	U	cpu	memory
www	0	fork	online	15	0.2%	66.3 MB

<pm2로 서버 구동 시킴>

5. 메인 게임 소스와 연결하기위한 socket 코드 작성

메인 게임은 C++을 사용하여 작성되었기 때문에 C++로 입력 받아 서버에 전송하기 위해서는 socket 프로그래밍이 필요하였다. socket 프로그래밍에 관해 공부한 뒤 필요한 부분에 대해 코드를 작성하였다. 따라서 터미널 상에서 유저가 입력한 정보를 받아 서버에 전달해서 필요한 정보를 받아올 수 있었다. 또한 유저 정보에 따라 게임이 종료되면 점수와 랭킹이 반환되어야 하므로 메인 게임 소스와 합치기 전 score라는 전역 변수를 두어 점수와 랭킹이 정상적으로 출력되는지를 확인하였다.

```

int sock, sign;
char buff[1000];

struct sockaddr_in serv_addr;

sock = socket(PF_INET, SOCK_STREAM, 0);

memset(buff, 0x00, sizeof(buff));
memset(myId, 0x00, sizeof(myId));
memset(myPwd, 0x00, sizeof(myPwd));
memset(&serv_addr, 0, sizeof(serv_addr));

serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = inet_addr("13.124.167.29");
serv_addr.sin_port = htons(3090);

write(sock, buff, sizeof(buff)); // send socket to server
memset(buff, 0x00, sizeof(buff)); // empty buffer
read(sock, buff, sizeof(buff)); // read socket from server

close(sock);

```

<socket 프로그래밍 핵심 부분>

<pre> -----LOGIN----- ID : sonbug PW : *** [1]signup [2]signin : 1 </pre>	<pre> sign up success !! -----LOGIN----- ID : PW : [1]signup [2]signin : 2 </pre>	<pre> -----LOGIN----- ID : sonbug PW : *** [1]signup [2]signin : 2 </pre>
--	--	--

```

sign up success !!

-----LOGIN-----

ID : sonbug
PW : ***

[1]signup [2]signin : 2

login success !

축하합니다. sonbug님이 최고점수 40점을 달성했습니다.
당신은 현재 3등 입니다. 1등의 390점 도전해보세요!

```

<socket 프로그래밍을 완료한 후의 view>

6. 로그인, 회원가입 시 유저 정보 예외 처리

같은 아이디의 중복을 방지하고 틀린 비밀번호가 입력됐을 때 서버가 정상적으로 작동하게 하기 위해 예외처리를 진행하였다. 유저의 정보가 입력되고 1, 2 중에 선택을 하면 서버에서는 반환되는 user_idx 값을 통해 어떤 유저인지 파악한다. 이때 조건을 걸어 DB에 user_id가 없다면 반환되는 user_idx 값을 -1, -2 로 두어 socket 상에서 if(user_id == -1) 이 걸리면 이미 존재하는 회원인 것으로 처리해주었다.

1) 이미 가입된 아이디로 회원가입을 요청 시

```

-----LOGIN-----

ID : sonbug
PW : ****

[1]signup [2]signin : 1

already exist user!

```

2) 틀린 비밀번호 입력 시

```

-----LOGIN-----

ID : sonbug
PW : ****

[1]signup [2]signin : 2

pwd error !

```

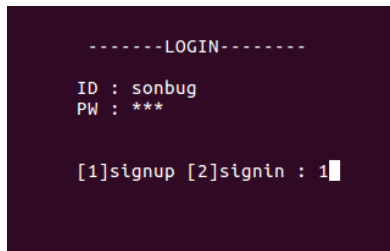
7. 비밀번호 암호화

프로그램에 회원가입, 로그인과 같은 유저 정보가 포함되어 있으므로 보안에 관한 부분이 필요하다고 생각했다. Node.js를 공부하는 도중 내장 모듈 중에 비밀번호 암호화와 관련된 "crypto" 모듈이 있다는 것을 알게 되었고 사용하였다. 해싱 알고리즘으로는 SHA512를 사용하였고 좀 더 확실한 보안을 위해 Salting + Key Stretching을 함께 사용하였다. 기존에 테이블에 존재하던 user_id와 더불어 user_salt가 필요하게 되었고 추가해줬다.

Result Grid						Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
user_id	user_pw	user_salt	user_idx	score					
yeji	OTt9Ur7vdOvMDLu4GleBTHagWKiFTHR1ZOMN...	IPjm482eKobrVxqKarc+ZUY1GIsW15FNCbXcRA84aY=	40	40					
ossp7	8cr8Ao/YK1E3067G01Bcv8zu3xGYRde2ZJriaTs...	SGeh5cxbr1yQEFF65TjBUOaCdrszdJ8pxoxyDOyE98=	41	40					
miranlee	oyhnPIgrNmbUk/ujBZV2Ta+W5SKomnz210RHX...	IFB5gCd0gOXUTwunVG23bSB131SkEQEU+aPS24gX/dw=	43	310					
hyex	WCcAR049vrJzBj4dUVk+HYg6BJVwA/eQP0PH...	AdzcwYaQscX8E8cM3S8hQQEQE0R4wvvyZF66q2vfmMLI=	44	390					
miran	qOIY8HWeUFZ3tv/1hl+Qhf9RBb5f8A1WHZVDJ...	M58nUgL1Su1kUPwJruEknHkMGJUJ+G3iINLAQUyL8=	50	40					
sun	1ny6CQOfvZiVJAOnVGrARRCyzwBAmlUP0udHs...	X59Xp/pv8fW6rxj8JVoH9I7xMjUgneT2fa05kymo4vo=	52	40					
sonbug	tmItzjghCp0y8Ch19P0ufJ6YDrBafG11WabYf3A...	+zsKMMttCr867v8Yd++XxYIjWDcSCMT54AWx7KwFnY=	55	200					

<password가 해싱되어 있고 user_salt 애트리뷰트가 존재함 확인>

데이터베이스에서의 암호화와 더불어 유저 view 상에서도 비밀번호 암호화를 진행하였다. 사용자를 제외한 타인이 비밀번호를 알 수 없게 하였다.



<password view 암호화>

8. Face tracker과 연결 후 최종 실행 파일 생성

모든 서버 동작, socket 코드가 정상적으로 작동함을 확인하고 메인 게임 소스와 연결하였다.

4.기대효과

<기대효과 및 활용분야>

게임으로서의 능력을 발전시키기 위해 여러 시도를 하여 피버타임, 장애물, 속도 등을 조절하여 사용자가 재미를 느낄 수 있도록 하였다. 따라서 이 게임을 통해 여러 사용자가 노트북으로 하는 페이스 게임을 느껴볼 수 있을 것이고, 새로운 시도이기 때문에 굉장히 흥미로울 것이다. 또한 주위 동기들과 게임을 해본 결과 모두 최고기록을 깨기 위해 노력했고, 별도의 장치 없이 얼굴만으로 게임을 진행하는 부분이 신선하다고 평가하였다.

향후 이 프로그램은 입 모양을 자세하게 인식할 수 있기 때문에 향후 청각 장애인을 위한 발화 프로그램으로 발전시킬 계획이다. 이에 대한 구체화된 아이디어는 다각형화되어 인식되고 있는 입모양을 아,에,이,오,우 특정 입모양과 일치시킨다. 이를 위해선 기존 FaceTracker의 코드에도 수정이 필요하지만 한학기동안 봐왔던 코드기에 어려움이 없을 것이라고 생각된다. 날아오는 도형대신 모음(ㅏ, ㅓ, ㅣ, ㅜ, ㅠ)이 날아오도록 한다. 이 때 입모양과 일치하는 모음을 먹어야만 점수가 올라간다. 예를 들어 ㅏ라는 도형을 먹고 싶으면 입모양도 “아”라는 모양을 하고 있는 상태에서 도형과 닿아야 점수가 올라간다. 이와 같이 게임을 통해 청각 장애인들이 즐겁게 발화를 할 수 있도록 도와주는 프로그램으로 발전시킬 수 있을 것이다.

<p>5.구현과정에서의 문제점 및 해결과정</p>	<p>위의 구현과정에서 문제점, 해결방안에 대해 자세히 서술하였지만 간단하게 문제점과 해결방안을 정리하면 다음과 같다.</p> <p><구현과정에서 있었던 문제점></p> <ol style="list-style-type: none"> 1. face tracker 오픈 소스 실행 이번 프로젝트에서 사용한 오픈 소스인 face tracker은 8년전에 개발된 오픈 소스로써 라이브러리나 해당 OS또한 옛날 버전을 사용하였고, 관련된 최신 자료가 없어 실행에 문제가 생겼다. 또한 웹 카메라를 실행시켜야 되나 virtual box는 바로 웹카메라가 지원이 되지 않았다. 2. server, Nodejs에 대해 지식, 개발 경험 무 server파트에 대해 지식이 있는 팀원이 없었다. 따라서 server 파트 구현의 어려움이 발생했다. 3. face tracker 코드 분석 이 오픈 소스의 파일 구성은 직관적으로 해석하기는 어려웠다. 따라서 파트를 나누어 코드 분석, 파일 구조를 분석하였다 4. face tracker에서 입 영역만을 인식 이 오픈 소스는 얼굴의 눈, 코, 입, 얼굴형을 모두 인식하는 소스이고, 위 팀에서는 입 영역을 인식하는 것이 중요했다. 따라서 입 영역을 어떻게 세밀하게 인식할 수 있는지를 파악해야 한다. 5. 물체 떨어트리기 물체를 화면에서 떨어지도록 구상하기 위해선 지금 include되어 있는 라이브러리 중 가능한 라이브러리가 없었다. 따라서 다른 방법이 필요했고 이를 해결해야했다. 6. OpenCV와 OpenGL의 배경 호환 문제 OpenCV와 OpenGL이 호환이 안되기 때문에 OpenGL로 그린 도형을 OpenCV를 배경으로 하여 출력하는 것에 어려움을 겪었다. 7. node, npm 설치 node js를 사용하기 위해 node, npm 을 설치하는 데 문제가 발생했다. 이미 버전이 있다고 나오나 사용이 되지 않거나, 설치하지 않은 상태에서 설치가 되지 않았다. 8. 좌표계 문제 OpenCV로 작성된 Facetracker을 OpenGL 로 변경하여 실행했을 때 어색함이 발생하였다. OpenCV는 모니터의 왼쪽위가 (0,0)로 x,y축 방향으로 양의 수만큼 증가하는 반면에 OpenGL은 모니터의 가운데가 (0,0)이었고 음의 방향으로 음의 수만큼, 양의 방향으로 양의 수만큼 증가하기 때문에 좌표계가 일치하지 않았다. 9. AWS, RDS 연결 로컬 서버만으로 개발을 하는 중, 여러 컴퓨터에서 사용할 수 있도록 서버를 여는 게 필요하여 아마존의 aws, rds를 사용하게 되었다. 계정을 생성하였으나 Nodejs와 이어주는 과정에서 문제가 발생했다.
-----------------------------	--

10. 랜덤하게 날아오는 물체 구현

물체가 날아오는 것에는 위에서 성공했지만 랜덤 한 방향에서 날라오도록 하는 데에는 더 많은 고민이 필요했다.

11. 웹카메라의 딜레이 문제

virtual box의 ubuntu에서 프로젝트를 진행하면서 웹카메라의 버퍼링이 심하게 발생하였다. 게임의 특성 상 딜레이가 발생하면 안되기 때문에 해결 방안을 찾아야 했다.

12. C++ 코드와 서버와의 통신

Nodejs 코드와 C++ 코드를 연결하기 위한 방안이 필요했다. 처음엔 라이브러리를 이용할까 했으나 사용에 어려움을 겪어 소켓 프로그래밍에 대해 도전하게 되었다.

13. 비밀번호 암호화

비밀번호를 암호화하여 RDS의 업데이트 하는 과정에서 어려움을 겪었다. crypto 모듈을 사용하여 salt 값으로 랜덤 한 값을 만들었으나 데이터베이스에 이상하게 올라갔다.

<구현과정에서의 해결과정>

1. 기존 팀원들은 각각 다른 virtual box version을 사용하고 있었다. 문제가 발생함에 따라 모두 downgrade를 하였고, ubuntu의 version 또한 16.04 version으로 변경하였다. 또한 필수였던 라이브러리 OpenCV의 3.2.0 version으로 설치하였다. 카메라 작동을 하기 위해 ubuntu의 카메라 패키지인 cheese를 실행하여 노트북의 내장 카메라가 인식되지 못함을 확인하였다. 따라서 virtual box의 extension pack을 추가 설치하여 vBoxManage를 통해 webcam을 수동으로 attach후 카메라 작동에 성공하였다.

2. node js 영상 강의를 수강하였고, 공부를 하였다. server에 대한 이해부터 node js의 기초부터 하나하나 직접 실행해보며 사용방법을 익혔다. 많은 시간을 소요했지만 일주일에 4시간씩 공부하는 시간을 가졌으며 이를 통해 프로젝트를 서서히 진행할 수 있었다.

3. 팀원 별 파트를 나누어 코드를 분석하였다. 각자 맡은 파트에 주석을 달아 팀원들과 공유하였고 파일 구조에 대해 추가 회의 시간을 통해 이해도를 높였다. 이에 어떤 함수를 사용하고 어떤 라이브러리를 사용할 지에 대한 느낌을 파악했다.

4. facetracker의 구성 요소 중 빨간 동그라미 부분이 계속해서 움직이는 영역을 포착했다. 해당 영역에 인덱싱을 추가하여 입 안 영역을 인식하기 위해서 필요한 값을 얻어냈다. 이는 바뀌지 않는 고유의 인덱스로써 입 영역 구분을 성공하였다.

5. OpenGL이라는 라이브러리를 설치했다. 우리가 사용한 오픈 소스인 facetracker은 makefile을 통해 컴파일 되는데 이 라이브러리 또한 컴파일 시 옵션이 필요하다. makefile이 아주 복잡한 구조를 이루어졌기 때문에 이 파일의 분석이 먼저 필요하였다. 분석을 통해 OpenGL 라이브러리를 포함시키는데 성공했

다. 이 라이브러리는 그래픽, 그림 등을 다룰 수 있는 라이브러리로, 우리는 기본적인 점, 선 면 표현이 가능함을 확인하고 이 라이브러리를 통해서 물체를 떨어트리는 것에 성공했다. 또한 OpenCV 라이브러리를 통해 원, 네모, 세모 등 원하는 모양을 생성하였고 이를 OpenGL을 통해 움직이도록 연결하였다 OpenGL을 지속적으로 공부하여 물체가 랜덤 한 속도와 랜덤 한 방향으로 떨어지게 구현하였다. 물체의 색깔 또한 랜덤 한 색으로 나오도록 하였다.

6. 처음에 고안한 방법은 OpenGL의 도형을 OpenCV화면에 변환하여 출력하는 것이었다. 하지만 구현을 마치고 보니 OpenGL의 애니메이션 기능을 사용하기 위해선 glutTimerFunc()함수가 필수불가결하였다. 따라서 OpenCV의 화면을 OpenGL의 배경으로 변환하는 것이 더 적절해보였다. 따라서 형변환을 위한 MatToTexture 함수를 작성하였고 텍스처 바인딩을 통해 OpenGL의 배경으로 포함시켰다. 또한 기존 main의 while(1)에 있던 코드들을 timer함수로 옮겨 1초마다 불러들이게 해 while문과 같이 작동하도록 하여 timer 함수로 main 무한루프에 있던 코드를 옮겼다. 그리고 기존 FaceTracker 코드는 키보드의 esc버튼만을 통해 종료할 수 있고, 키보드의 d를 입력 받아 frameReset이 가능하였다. glutKeyboardFunc을 이용하여 keyboard의 signal이 오면 해당 입력 값을 받아 실행시키도록 하여 기존 FaceTracker와 완전히 동일한 기능을 수행하도록 OpenGL 코드로의 변환에 성공하였다.

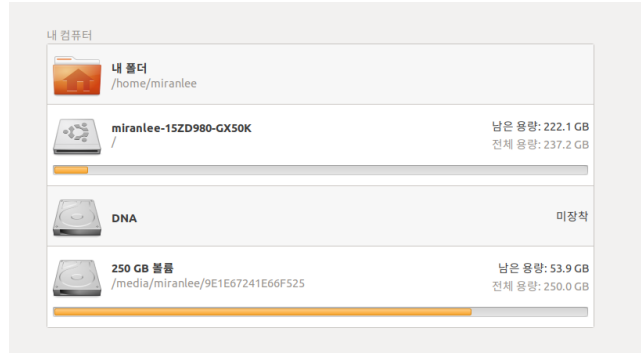
7. node js는 여러 폴더로 설치가 진행되어 삭제 시 어려움이 발생했다. 여러 버전이 존재하고 버전에 따라 사용할 수 있는 모듈의 제한이 있었다. 따라서 모두 재설치를 진행하고 버전 관리가 가능한 nvm을 설치하여 버전을 8.9.34로 유지하였다.

8. 원의 크기와 gluOrtho2D 함수를 조정하여 어느 정도 조정은 했으나 좌표계에 대한 이해가 필요했다. 관련 서적을 참고하여 사용가능한 함수와 방법에 대해 고안했고 이를 통해 해결하였다. CVtoGL 함수를 구현하여 OpenCV의 좌표계로 인식된 입, 얼굴의 위치를 적절한 식을 통해 계산하여 OpenGL 좌표계와 일치시켰다.

9. 데이터베이스로는 MySQL을 사용했으며, 이를 rds에서 이용하는 것을 확인하기 위해 MySQL workbench를 설치했다. 서버로 값이 넘어가고 작동하는 것을 보기 위해 postman을 통해 확인했다. 먼저 포트 문제가 발생했다. 아예 작동에 실패했고, rds의 보안 그룹에 우리가 사용할 포트를 추가하여 이를 해결했다.

10. 날아오는 도형 함수 구현은 랜덤 한 수를 생성하여 상하좌우, 랜덤 한 곳에서 색상을 변경하며 도형이 날아오게 하였다. 좌표 값을 계산하여 도형이 상, 하, 좌, 우 중 어느 한곳과 부딪히면 사라지게 하였다. 계속해서 Timer함수를 호출하여 끊임없이 나오도록 하여 종료하지 않도록 하였다. 해당 함수와 FaceTracker를 결합하여 FaceTracker 화면에 랜덤으로 날아오는 도형을 출력할 수 있었다.

11. Window10의 linuxSubSystem으로 환경을 교체하였지만 해당 환경은 외부기기를 지원하지 않아 웹카메라의 사용이 불가능하였다. 교수님과의 면담을 통해 외부 장치에 가상머신없이 Ubuntu를 설치하라는 조언을 얻었고 ssd를 업그레이드하여 256GB의 추가 용량을 partition으로 나누어 전부 Ubuntu로 설치하였다. 작동 결과 버퍼링이 존재하지 않는 것을 확인하였다.



12. 우선 c++ 코드를 gcc 컴파일러로 컴파일할 수 없었다. 따라서 컴파일

명령어에 -lstdc++ 를 추가하여 컴파일에 성공했으나 소켓 연결에 실패했다. 포트의 연결이 꼬임으로 인해 발생한 문제였고, 코드 수정을 통해 해결했다. 그러나 RDS에 저장된 회원 정보로 로그인을 하는데 계속하여 실패했다. 서버 코드와 C++ 코드를 계속하여 확인하였지만 문제를 찾을 수 없었다. 여러 시도를 계속해보다가 회원가입부터 소켓을 통해 진행했더니 로그인에 성공했다.

13. crypto 모듈을 통해 랜덤 한 값을 생성하고 Salt값을 database에 올리는 거까지 성공했지만, 비밀번호 값이 이상하게 올라갔다. 데이터베이스로부터 비밀번호를 꺼내 다시 복호화를 하고 Salt값을 통해 암호화를 했을 때 동일한 값이 나와야 하는데 다른 문제가 없음에도 계속하여 불일치가 발생했다. 계속 여러 코드를 수정해본 결과 데이터베이스의 문제라는 것을 발견했다. 우리가 사용한 암호화 방식은 총 32자리의 암호를 만드는 방식인데 데이터베이스에서 들어갈 수 있는 범위가 30자리 수까지였기 때문이다. 따라서 데이터베이스를 수정하여 문제를 해결했다.

<p>6. 참고문헌</p>	<p>참고 사이트</p> <ul style="list-style-type: none"> • Face Tracker 오픈 소스 https://github.com/kylemcdonald/FaceTracker • OpenGL 가이드 https://monochromelux.github.io/opengl/ • OpenCV와 OpenGL 변환 https://babytiger.tistory.com/entry/OpenGL%EB%A1%9C-%EA%B7%B8%EB%A6%B0-%EC%9E%A5%EB%A9%B4%EC%9D%84-OpenCV-%EC%9C%88%EB%8F%84%EC%9A%B0%EC%97%90-%ED%91%9C%EC%8B%9C <p>참고 서적</p> <ul style="list-style-type: none"> • (OpenGL로 배우는)컴퓨터 그래픽스-주우석(한빛미디어) • OpenCV로 배우는 영상 처리 및 응용-정성환(생능출판사)
----------------	---