캡스톤 설계 최종 보고서

Face Converter

프로그램 명 : Face Converter
제 작 자 : ○○○○○○○ 김○○
제 작 기 간 : 2022년 학기 중 (6개월)

목차

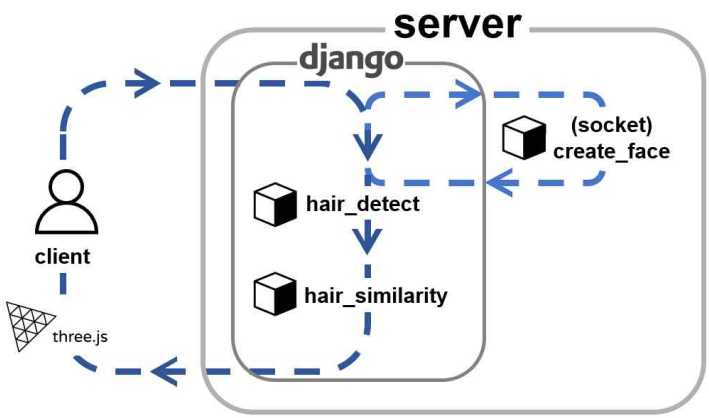
1. 소개

사진속의 인물의 얼굴을 3차원 시각화 프로젝트입니다.
최근 메타버스에 대한 주목과 하드웨어의 발전으로 3차원 이미지에 대한 접근성과 관심도가 높아짐에 따라 3차원 객체를 다루는 것에 관심이 생겨 프로젝트를 기획하였습니다.

간단한 UI를 조작하여 번역기, 파일 확장자 변경 서비스를 제공하는 웹사이트처럼, 2차원 이미지를 복잡한 모델링을 사용하지 않고서도 개인만에 3D 이미지로 변경하는 서비스를 제공하는 웹 프로젝트 입니다.

원래는 앱(응용프로그램)프로젝트로 기획하였으나, 프로젝트의 가장 중요한 모듈의 문제 발생으로 웹프로젝트로 변경하였습니다.

1.1) 프로젝트 흐름



Django(파이썬 웹 프레임워크)와 소켓을 활용하여 서버를 구성하였고, 서버 내부에는 프로그램 모듈들을 배치하였습니다.

프로그램의 작동 순서는 다음과 같습니다.

<1> 사용자의 2차원 이미지를 입력받음.
<2> 사용자의 이미지 필터링.
<3> create_face : 사용자 이미지를 활용하여 얼굴에 대한 이미지 3차원 이미지 생성.
<4> hair_detect : 사용자의 이미지에서 헤어 영역을 검출.
<5> hair_similarity : 검출한 헤어 영역과 가장 유사한 3D 헤어모델 매칭.
<6> 사용자에게 3D 모델 전송

(create_face 모듈에 사용되는 bpy라이브러리로 인하여 응용프로그램으로 묶을 수 없고, 또한 django 내부에서 사용 시 제대로 동작하지 않아. 소켓 프로그래밍을 이용하여 해당 문제를 해결.)

2. 개발 과정

프로젝트의 주요 모듈 개발 과정입니다.

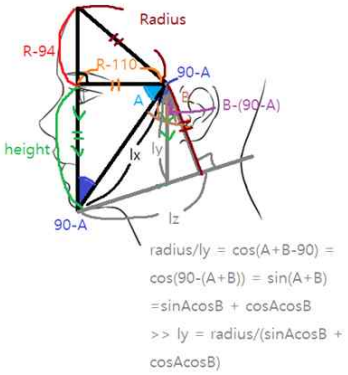
1. create_face
2. hair_detect
3. hair_similarity

2.1) create_face

해당 모듈에서는 3차원 객체를 다루는 블렌더를, 파이썬 내부에서 조작하는 bpy(**b**lender + **py**thon) 라이브러리를 사용하였습니다.

2.1.1. dlib 활용

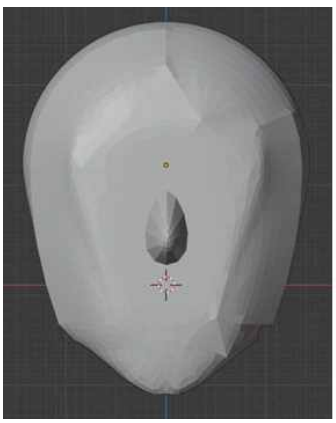
2차원 이미지를 3차원 이미지로 변경해주는 모듈입니다. 초기에는 dlib를 이용하여 얼굴의 68개의 랜드마크를 활용하여 해당 랜드마크를 영상처리를 활용하여 영역을 추출하고 추출한 영역을 기반으로 삼각함수 등 다양한 계산을 통해 3D 모델을 제작하려고 기획했으나 해당 결과물이 좋지 못하여 다른 방법을 찾아보았습니다.

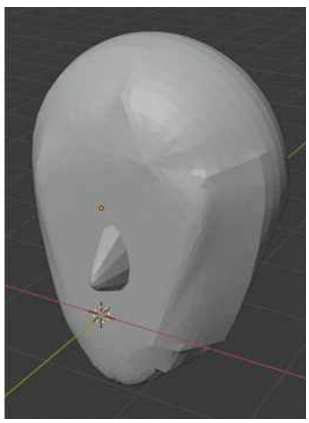


삼각함수를 이용한 모델 설계

dlib 68개의 랜드마크 추출

영상처리를 활용한 영역 추출





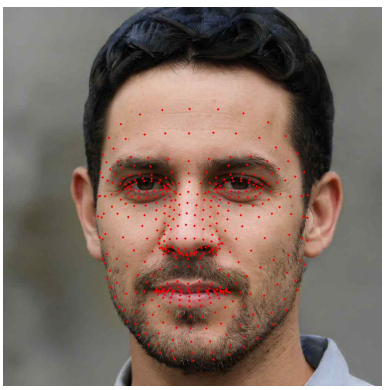
위의 방법을 활용한 결과입니다.

68개의 랜드마크를 활용하여 3D 모델을 제작하기에는 보다시피 결과가 좋지 않습니다.

2.1.2. mediapipe 활용

mediapipe 라이브러리의 face_mesh를 활용하면 dlib를 사용하는 것보다 더 많은 468개의 랜드마크를 구할 수 있고, 랜드마크는 x,y 좌표 뿐만 아니라 z좌표까지 제공하기 때문에 mediapipe를 활용하였습니다.

랜드마크와 bpy를 활용하여 점들을 위치를 조정하면 얼추 얼굴의 모습이 보입니다.



mediapipe.face_mesh 좌표 2차원

mediapipe.face_mesh 좌표 3차원

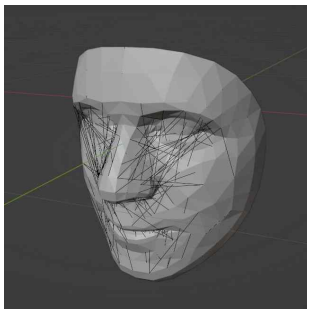
2.1.3. 안면 객체 제작

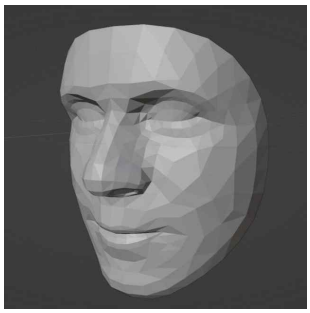
mediapipe의 468개의 랜드마크의 점들을 이용하여 면을 제작하였고, 좌우대칭, Shade Smooth(표면 매끄럽게)를 적용하여 안면 객체를 제작하였습니다.

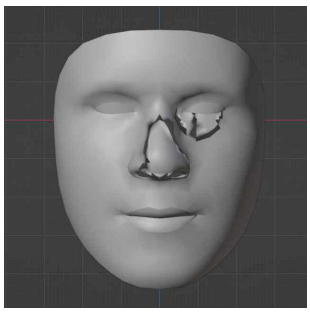
점3개 혹은 4개를 묶어서 면으로 제작하였습니다. 한면을 구성하는 인덱스를 구하는 것은 자료가 없어서 직접 추척하며 만들어야 했습니다.

좌우대칭또한 점들의 인덱스를 직접 추적하여 매칭 점들을 구하였습니다.

블렌더에서는 점 혼자 내부에 존재할 수 없기 때문에 점들의 집합 468개의 면을 생성하였고, 이후 면을 채운후, 기존에 필요 없는 생성시에 생긴 선들을 제거 하였습니다.


면 채우기 결과


선 제거 결과


Shade Smooth 문제 발생


Shade Smooth 문제 해결

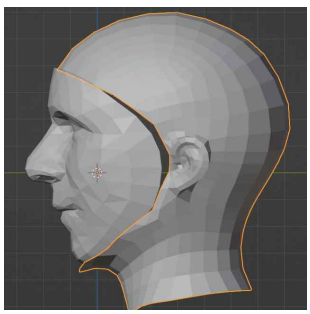
Shade Smooth는 표면을 매끄럽게 하는 기능인데 이를 적용하면 특정 부위의 면이 뒤집히는 현상이 발생하여 모든 면들의 방향을 재정렬 하는 기능을 활용하여 해당 문제를 해결하였습니다.

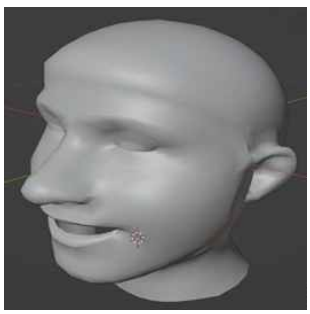
2.1.4 두상 붙이기

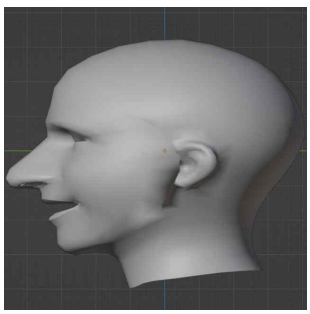
사람에 안면에 대해서는 정보를 얻을 수 있지만 후면에 대해서는 정보를 얻을 수 없기 때문에 임의로 두상 모델을 만들어서 이를 안면모델이랑 병합하는 방법으로 진행하였습니다. 두상모델은 2번의 수정을 하였습니다.

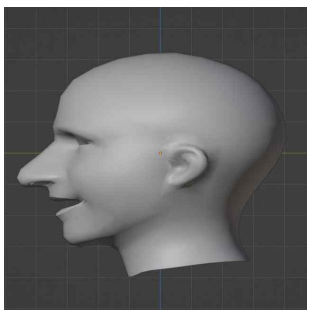
안면과 두상모델의 턱끝을 기준으로 사이즈 조정, 병합을 하였고, 이에 특정 케이스에서는 이마꺼짐 현상이 발생하여, 사이즈조정, 병합전에 두 객체의 회전을 적용하여 문제를 해결하였습니다.

또 병합중에 특정 케이스에서는 귀~턱선 쪽에 그림자가 생기는 현상이 발생하였고, 원인을 분석한 결과 객체의 점들이 몰려서 발생한 현상이었습니다. 점들을 분산하여 해당 문제를 해결하였습니다.


두상모델 3차

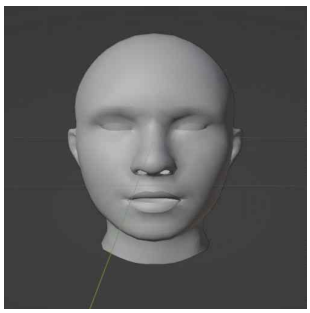

이마꺼짐 현상

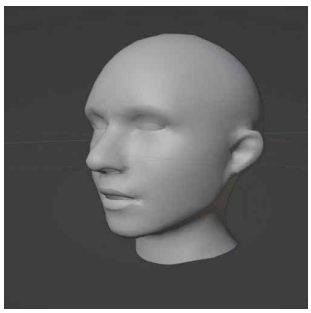

그림자 현상 발생

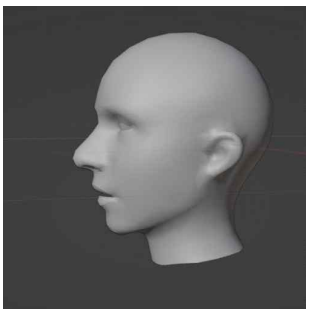

그림자 현상 해결

2.1) create_face 의 최종 결과입니다.


사진 원본







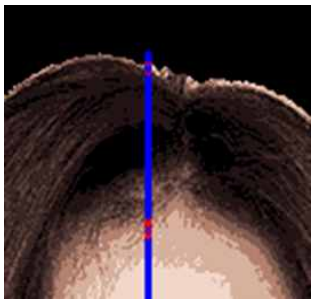
2.2) hair_detect

헤어부분에 대한 모델은 이미지에 따라 제작할 수가 없다고 판단하여 유사한 헤어모델을 매칭하는 방법을 사용하였습니다. 이 방법을 사용하기 위해서는 헤어의 영역을 검출해야 합니다.

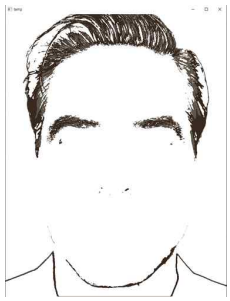
2.2.1) 알고리즘 방식 접근

처음 헤어영역 검출은 알고리즘 방식으로 접근하였습니다.

(1) 이미지의 컬러 수를 줄이고 이마 위쪽방향으로부터 탐색하여 픽셀의 값이 급격하게 변화하는 구간을 헤어 영역의 시작이라 판단하여 해당 픽셀을 가지는 영역 검출하는 방식

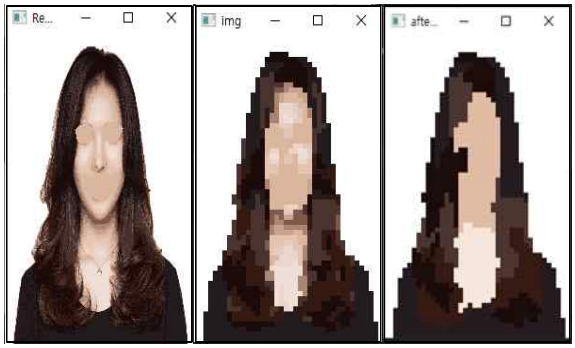


(1)번 방식의 과정



결과

(2) 헤어 영역의 불필요한 눈썹, 눈, 코, 입을 제거하고, 이미지의 컬러수, 크기를 줄이여 작은 영역들을 만들고, 영역들끼리의 거리, 픽셀 값을 사용하여 영역을 검출하는 방식

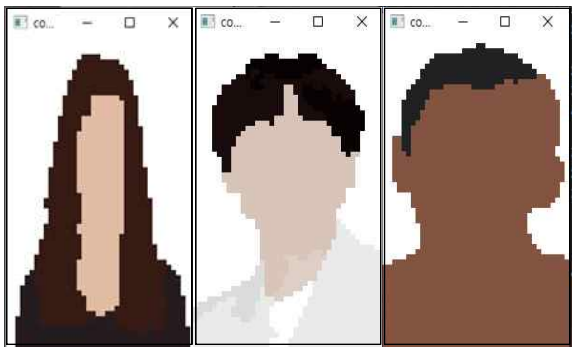


(2)눈코입제거 (2)그룹 나누기 (2)그룹 병합

등 다양한 방법을 시도해 보았습니다.

(2)번 방식은 특정 케이스에서는 어느 정도 만족하는 결과를 보여주었지만 특정 케이스에서는 제대로 검출하지 못하는 모습을 보입니다.

어느 정도 적당한 가중치를 찾았지만, 사람의 피부의 색, 헤어 색상에 성능이 변화기 때문에 알고리즘만을 의존하는 방식으로는 한계가 명확하기 때문에 방법을 달리 해보았습니다.

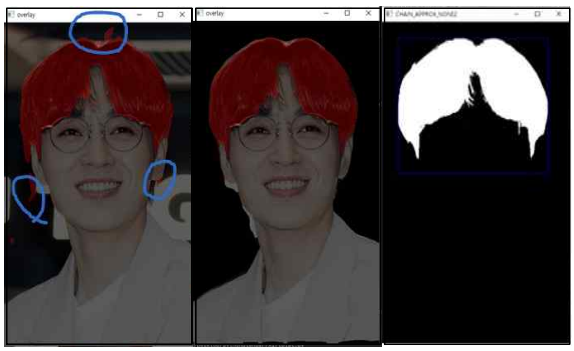


결과 예시1 결과 예시2 결과 예시3

2.2.2) 인공지능사용

인공지능과 Opencv의 영상처리를 활용하여 헤어영역을 검출하는 단계는 다음과 같습니다.

(1) 헤어 영역 검출
(2) 배경 제거
(3) 노이즈 제거



헤어검출 배경제거X 배경제거 노이즈제거

(1) tenseorflow, Keras 기반 DeepLabV3+를 활용한 20개의 영역을 검출하는 인공지능에서 헤어 영역만을 검출하도록 인공지능을 출력을 수정하였다.
(2) (1)과 마찬가지로 Pythorch 기반 DeepLabV3를 활용한 여러 영역을 검출하는 인공지능에서 배경만을 검출하도록 출력을 수정하고, 마스킹을 이미지를 생성하였다.
(3) Opencv의 이미지의 이진화, 경계선 검출을 활용하여 제일 큰 영역을 기준으로 특정 수치이상의 크기를 가지는 영역만 헤어 영역으로 판단하고, 이외의 영역을 제거하여 노이즈를 제거하였다.

2.3) hair_similarity

2.2)에서 검출된 영역을 기반으로 가장 유사한 헤어를 찾는 모듈입니다.

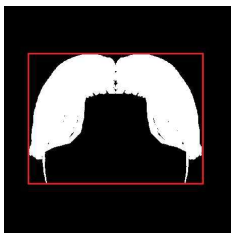
헤어 조합 리스트 정면 이미지(총 44개의 조합, 좌우 대칭까지 하면 74종류)

가장 유사한 헤어를 찾기 전에 헤어 모델이 필요하여 blender를 사용하여 헤어 모델의 라이브러리를 제작하였고 이에 대한 정면이미지들과 비교하는 모듈을 제작하였습니다

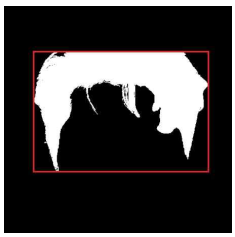
2.3.1) 헤어 매칭 알고리즘

compare image : 라이브러리 중 하나의 모델의 정면이미지

base image : 2.2)의 과정에서 추출된 헤어 영역



compare image



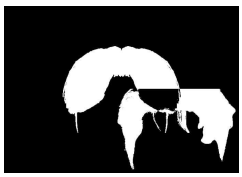
base image

알고리즘의 순서는 다음과 같습니다.

(1) 2.2)처럼 헤어 모델의 compare image도 전처리를 진행하고 영역을 구합니다.

(2) compare image와 base image를 비율을 유지하면서 너비의 길이를 조정

(3) 좌측상단 위치

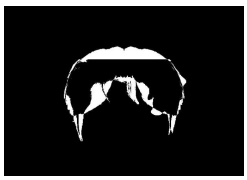


(4) 우측하단 종료

(3) 큰 컨버스를 생성하여 compare image를 중심에 base image 좌측 상단에 위치

(4) base image는 정해진 픽셀 수 만큼 우측으로 이동, 이동시 compare image와 base image에 대하여 이미지 비트연산 bitwise_xor 연산 진행, base image가 우측 끝으로 이동할 때 까지 반복

(5) base image를 다시 좌측 끝으로 보내고 하단으로 정해진 픽셀 수만큼 한 번 이동한다.

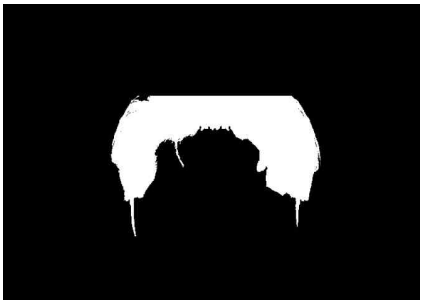


bitwise_xor 최소값

(6) (3)~(5)과정을 base image가 우측 하단에 위치할 때 까지 반복한다.

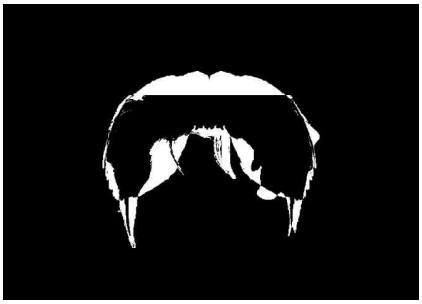
(7) 가장 bitwise_xor값이 가장 작을 때, 가장 많이 겹치는 시점을 기록하여, 모든 compare image이미지에 대해서 (1)~(6)과정을 진행한다.

(8) 모든 compare image에 대해서 bitwise_xor의 최소값 위치에 대하여 bitwise_and, bitwise_or의 이미지에 대해서 평균 제곱 오차(MSE, Mean Squared Error)를 구하여 최소값을 가지는 compare image를 구한다.



bitwise_and 예시

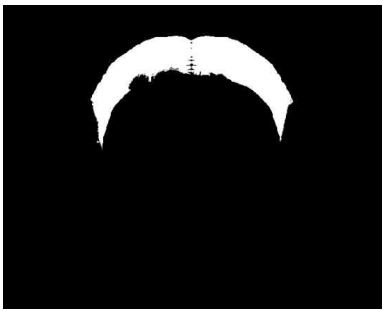
bitwise_or 예시



bitwise_xor 예시

bitwise_and 예시

bitwise_or 예시



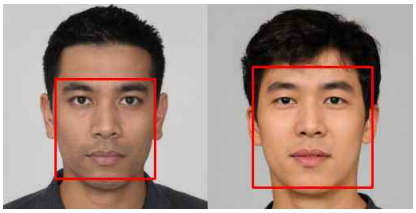
bitwise_xor 예시

위의 경우는 어느 정도 유사한 헤어모델의 경우, 하단의 경우는 극단적으로 유사하지 않는 헤어모델의 경우.

2.4) 그 외

2.4.1) 이미지 필터링

2.1) create_face 과정을 진행하기 전에 이미지 필터링이 필요하다. mediapipe의 face_mesh는 한 사람의 얼굴만 인식하여 사용자에게 받은 이미지에 대해서 dlib를 사용하여 필터링을 적용하였다.



dlib 활용 사람이 1명 아닌 경우를 걸러냄

2.4.2) 두상 헤어모델 병합

2.1) create_face 의 추출물과 2.3) hair_similarity 과정을 통해 얻은 헤어 모델의 정보를 이용하여 모델을 병합한다. (2.3의 결과물에 대한 정보는 헤어모델의 구성들의 묶음이다.)

헤어모델의 구성파일들을 읽어서 헤어모델을 불러오고 2.1의 결과 얼굴 모델에 대하여 사이즈 적용하면 다음과 같은 문제가 발생한다. 2.1과정에서 두상모델을 회전하고 사이즈를 조정하여 좌표들이 조금씩 이동하여 헤어 모델이 두상모델에 정확하게 위치하지 못하는 문제이다

사이즈만 조정 시 문제 발생

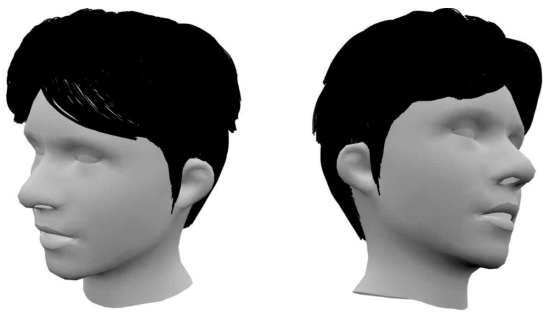
이를 해결하기위해 헤어 모델의 위치들을 두상모델에 특정 좌표로 이동시켜서 적절한 위치를 찾아 배치하였다.

사이즈만 조정, 좌표 수정 후 결과

2.4.3) 클라이언트 화면 출력

웹에서 3D객체를 출력할 때에는 THREE.js를 사용한다. THREE.js에서 GLTFLoader를 사용하여 3D객체를 출력한다. 파일 형식은 3D 파일의 확장자인 .fbx .gltf 보다 읽는 속도가 빠른 .glb 형식을 사용한다.

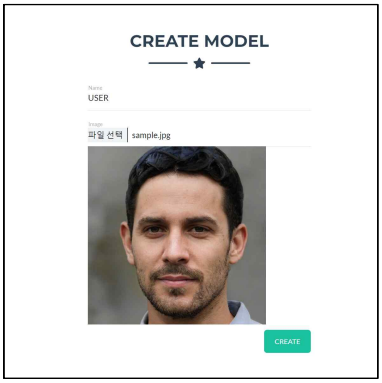
우측은 THREE.js GLTFLoader를 이용한 프로젝트의 결과물 샘플이다.

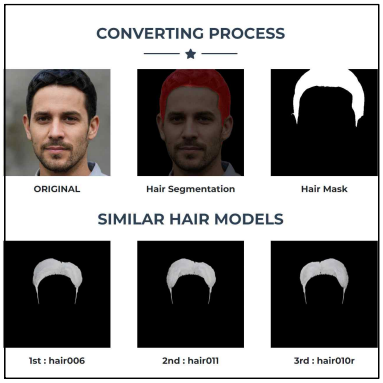


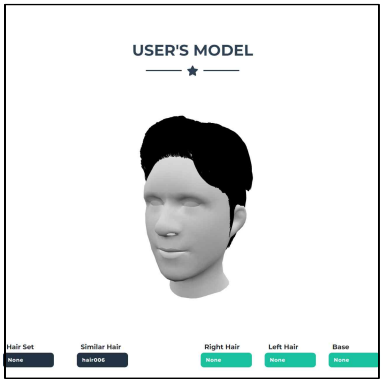
웹에서 THREE.js를 활용한 출력 결과 예시

3) 마무리

3.1) 시스템 구현과 동작 모습







3.2) 문제점, 한계

- 헤어모델의 용량이 너무 커서 읽는데 속도가 꽤 걸린다. 이를 해결하기 위해 헤어 모델의 퀄리티를 저하시켰지만 마찬가지로 긴 시간이 소요된다.

- 해당 웹 프로젝트를 AWS를 이용하여 배포하려 했으나, 헤어 모델의 용량과, 프로그램 동작의 어느 정도 수준의 하드웨어 성능을 요구하여 비용적인 문제로, 포트포워딩을 활용하여 배포하였다.

- 468개의 랜드마크로는 더 구체적인 얼굴 모델을 제작하기에 한계가 있다.