# SOLIDRAW

도 면 작 성 의 새 로 운 패 러 다 임

졸업은하겠조 | 김도영 신동호 손다연

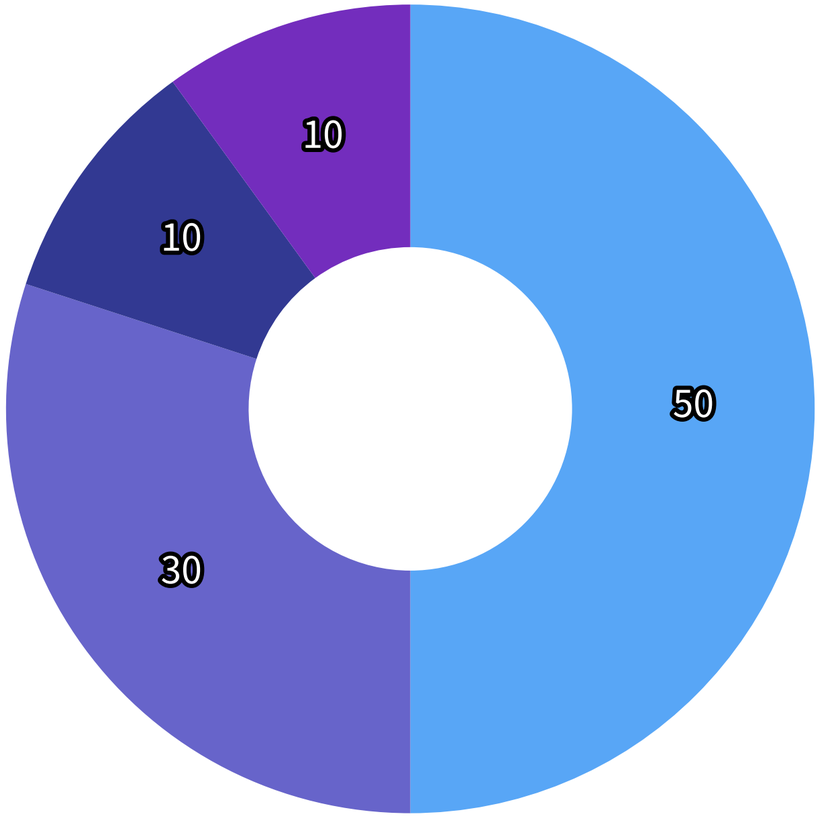
# CONTENTS

# 01

## SOLIDRAW

01 프로젝트 배경

# 건축 도면 설계의 어려움 | 건축과 설문




- 반복해서 도면을 작성하는 데 피로감을 느껴요 (50%)
- 도면을 작성하는 데 있어서 건축 법규, 아이콘 규칙 등 고려 할 사항이 너무 많아요 (30%)
- 다양한 프로그램을 독학해야 하고, 팍팍한 스케줄 때문에 힘들어요 (10%)
- 잦은 크리틱과 교수님께 거절당하는 게 힘들어요 (10%)

**손 스케치가 디지털 도면으로 자동 변환되면 좋겠다! (95.2%)**

03 프로젝트 내용

# 도면 작성의 자동화

SOLID

+

DRAW

### 기초 손스케치를 입력값으로 받는다

선과 치수, 공간의 쓰임만 기록된 기초 손 스케치를
입력 값으로 받는다.

### 프로그램의 도면 보정

프로그램은 손스케치를 직선으로 자동 보정
고정적 내부 인테리어를 자동 배치.
벽, 문 등 구조물을 건축 규정에 맞춰
평면도 기호로 변환한 후 도면에 치수 자동 기입.

**SOLIDRAW : 깔끔하고 규정을 준수한 도면 변환 프로그램**

# 02

## 구현 내용

01. 핵심 기능
02. 주요 코드
03. GUI
04. 진행

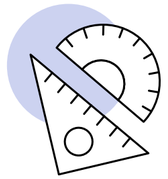
01 핵심 기능

# 손 도면을 깔끔한 도면으로 변환

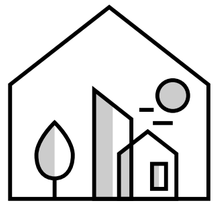
핸드 드로우를

SOLIDRAW로!

1. 기초 도면 직선 보정: 스케치 노이즈 제거, 경계선 검출, 경계선 직선 변환
2. 공간 구분: 도형 내부 도형 검출, 도형 사이 직선 활용하여 공간 구분, 영역 할당
3. 문자 및 숫자 인식, 공간 용도 판단: 공간 메모 인식, 공간 용도 판단
4. 내부 구조물 인식, 평면도 기호 변환: 설계 규정에 맞는 기호로 변환 및 치수 기입
5. 실내 인테리어 배치: 공간의 용도에 맞게 인테리어 배치.

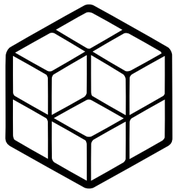


직선 보정

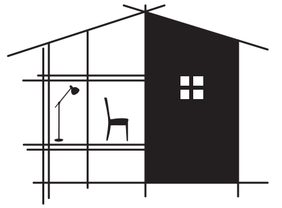


공간 구분

문자 인식



구조물 인식
기호 변환



인테리어 배치

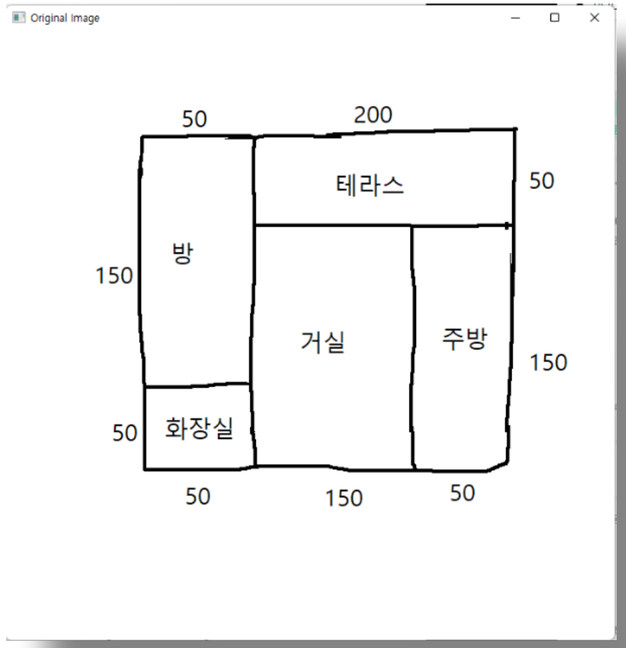
## 01 핵심 기능

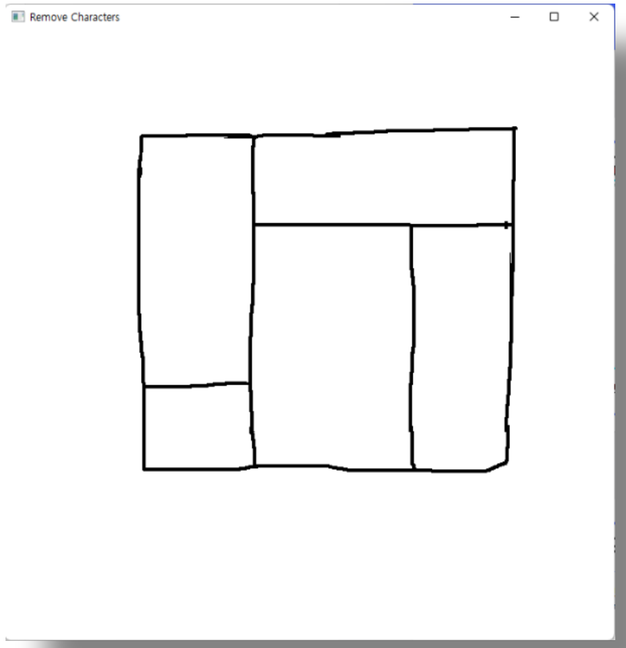
# 도면 변환 과정 - 1

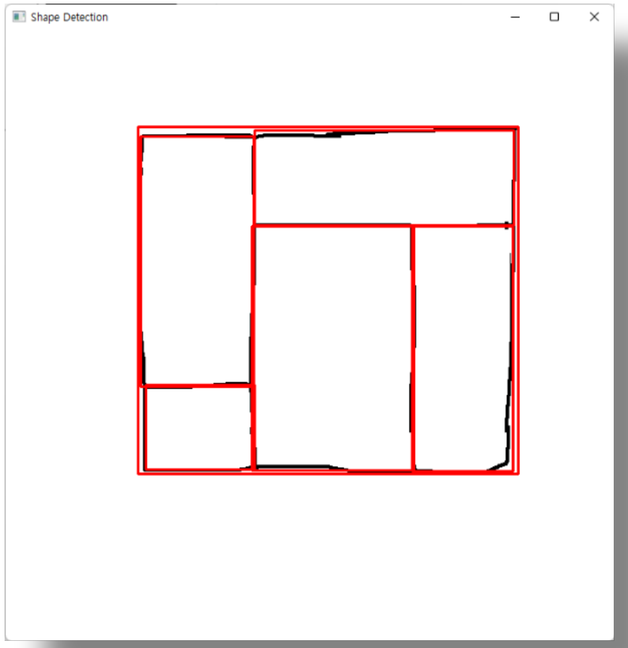
원본 이미지 업로드

문자 인식, 저장, 제거

구조 탐지




** 위 이미지는 동작 과정을 보이기 위한 출력 화면으로, 실제 작품에서 보이지는 않음*

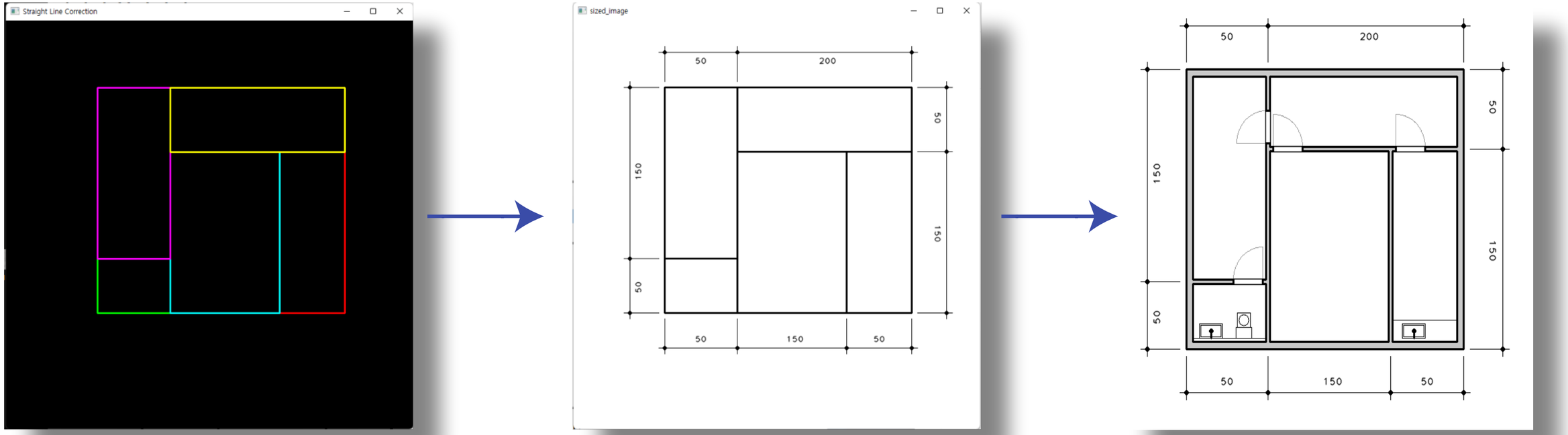
## 01 핵심 기능

# 도면 변환 과정 - 2

직선 보정

치수 입력

인테리어 배치




** 위 이미지는 동작 과정을 보이기 위한 출력 화면으로, 실제 작품에서 보이지는 않음*

## 02 주요 코드

# 원본 이미지 업로드

### Python Flask로 웹 서버 구현

```
#파일 업로드 동작
@app.route('/', methods=['POST'])
def upload_files():
    uploaded_file = request.files['file']
    filename = secure_filename(uploaded_file.filename)
    if filename != '':
        file_ext = os.path.splitext(filename)[1]
        #파일 확장자가 jpg, png, gif가 아니면 400번 에러 뱉음
        if file_ext not in app.config['UPLOAD_EXTENSIONS'] or \
                file_ext != validate_image(uploaded_file.stream):
            return "Invalid image", 400
        #다른 에러 처리기에 걸리지 않으면 input 이미지를 before.png라는 이름과 확장자를 가진 파일로 저장
        uploaded_file.save(os.path.join(app.config['UPLOAD_PATH'], "before.png"))
        #사진 업로드를 하면 convert.py를 따르는 기능 수행
        convert.main()
    #서버가 요청을 성공적으로 처리했지만 콘텐츠를 제공하지 않는다는 204 에러 뱉어냄
    return '', 204
```

1. 사용자가 웹을 통해 Input 이미지 등록 동작 실행
2. 서버는 Input 이미지를 request를 통해 받음
3. 이미지가 유효한 파일이라면 예외처리 시작
   3.1. 확장자, 또는 파일 크기가 프로그램과 맞지 않으면
   에러 처리기를 통한 예외처리 수행
4. 예외처리를 거친 뒤 적합한 파일임을 확인하면,
   'static/img' 라는 상대 경로 안에 'before.png'라는
   이름과 확장자를 가진 파일로 서버 폴더에 이미지 저장

## 02 주요 코드

# 문자 인식, 저장, 제거

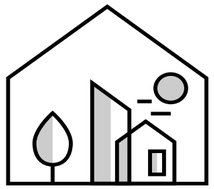
### Kakao OCR API 사용

```
# OCR 함수 호출
    rest_api_key = 
    output = kakao_ocr(image_path, rest_api_key).json()
    output_data = json.dumps(output, ensure_ascii=False, sort_keys=True, indent=2)

    # 받은 데이터 array로 변환
    output_data = json.loads(output_data)

    OCR_room = []
    OCR_size = []
    # 읽어 들인 문자 및 좌표 저장
    for i in range(len(output_data['result'])):
        x1 = output_data['result'][i]['boxes'][0][0]
        y1 = output_data['result'][i]['boxes'][0][1]
        x2 = output_data['result'][i]['boxes'][1][0]
        y2 = output_data['result'][i]['boxes'][2][1]
        recognition_words = output_data['result'][i]['recognition_words'][0]

        if recognition_words.isdigit():
            OCR_size.append([recognition_words, int((x1 + x2) / 2), int((y1 + y2) / 2)])
        else:
            OCR_room.append([recognition_words, int((x1 + x2) / 2), int((y1 + y2) / 2)])

        # 읽어 들인 문자 도면에서 제거
        for j in range(x1 - 1, x2 + 1):
            for k in range(y1 - 1, y2 + 1):
                image[k][j] = [255, 255, 255]
```

1. 서버 디렉토리에 업로드 완료된 이미지를 받아옴
2. 읽어온 이미지의 문자를 한글과 숫자 인식이 가능한 OCR 딥러닝 카카오 API를 활용하여 치수와 거실, 주방과 같은 공간 메모를 추출
3. 문자가 인식되면 치수는 OCR_size[]에, 공간 메모는 OCR_room[]에 저장
4. 저장된 문자들은 도면 직선 보정에 방해가 될 소지가 있으므로 인식된 문자의 좌표를 기준으로 문자를 도면에서 제거

** api 키는 개인에게 발급된 고유 키이기 때문에 본 발표 자료에서 제거(rest_api_key)*

## 02 주요 코드



# 공간 구분 | 직선 보정

```
# 이미지 흑백 변환 및 이진화
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY_INV)
# 외곽선 검출
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

# 검출된 모든 도형 출력
for c in contours:
    x, y, w, h = cv2.boundingRect(c)
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)
    # cv2.imshow('Shape Detection', image)
# cv2.waitKey(0)

# 도형 개수 카운트 변수
cnt = 0
# 도면의 모서리 끝 좌표
upper_left = [0, 0]
upper_right = [0, 0]
lower_left = [0, 0]
lower_right = [0, 0]
# 도면 선 색상 리스트
colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255, 255), (255, 255, 255)]
# 각 공간별 접선 좌표 통일을 위한 좌표 저장 리스트
points_x = []
points_y = []

# 새로 기입할 치수 위치
size_points = []
for i in range(len(OCR_size)):
    size_points.append([])
# 각 방의 보정 좌표 값 저장
rooms_points = []
```

Python OpenCV의 cvtColor, threshold, findContours 함수 사용

1. cvtColor와 threshold 함수를 사용하여 이미지를 흑백 변환 및 이진화
2. findContours 함수를 사용하여 얻은 각 도형의 꼭짓점을 기준으로 도형 검출
3. 각 공간을 구분
4. 각 공간과 공간 사이에 빈 여백이 없도록 도형 배치 조정
5. 도면 직선 보정 완료

## 02 주요 코드

# 치수 및 치수선 기입

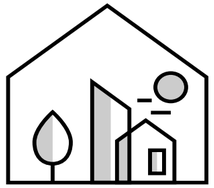
```
# 치수 기입
    font = cv2.FONT_HERSHEY_PLAIN
    for i in range(len(size_points)):
        text = str(OCR_size[i][0])
        if len(text) < 4:
            text = ' ' + text

        if size_points[i][0][0] < upper_left[0]:
            result_image = cv2.rotate(result_image, cv2.ROTATE_90_CLOCKWISE)  # 시계 방향으로 90도 회전
            sized_image = cv2.rotate(sized_image, cv2.ROTATE_90_CLOCKWISE)  # 시계 방향으로 90도 회전
            points = (sized_image.shape[0] - size_points[i][0][1] - 22, size_points[i][0][0] + 30)
            result_image = cv2.putText(result_image, text, points, font, 1, (0, 0, 0), 1, cv2.LINE_AA)
            sized_image = cv2.putText(sized_image, text, points, font, 1, (0, 0, 0), 1, cv2.LINE_AA)
            result_image = cv2.rotate(result_image, cv2.ROTATE_90_COUNTERCLOCKWISE)  # 반시계 방향으로 90도 회전
            sized_image = cv2.rotate(sized_image, cv2.ROTATE_90_COUNTERCLOCKWISE)  # 반시계 방향으로 90도 회전
        elif size_points[i][0][0] > upper_right[0]:
            result_image = cv2.rotate(result_image, cv2.ROTATE_90_COUNTERCLOCKWISE)  # 반시계 방향으로 90도 회전
            sized_image = cv2.rotate(sized_image, cv2.ROTATE_90_COUNTERCLOCKWISE)  # 반시계 방향으로 90도 회전
            points = (size_points[i][0][1] - 21, sized_image.shape[1] - size_points[i][0][0] - 10)
            result_image = cv2.putText(result_image, text, points, font, 1, (0, 0, 0), 1, cv2.LINE_AA)
            sized_image = cv2.putText(sized_image, text, points, font, 1, (0, 0, 0), 1, cv2.LINE_AA)
            result_image = cv2.rotate(result_image, cv2.ROTATE_90_CLOCKWISE)  # 시계 방향으로 90도 회전
            sized_image = cv2.rotate(sized_image, cv2.ROTATE_90_CLOCKWISE)  # 시계 방향으로 90도 회전
        else:
            points = (size_points[i][0][0] + 2, size_points[i][0][1])
            result_image = cv2.putText(result_image, text, points, font, 1, (0, 0, 0), 1, cv2.LINE_AA)
            sized_image = cv2.putText(sized_image, text, points, font, 1, (0, 0, 0), 1, cv2.LINE_AA)
```

```
# 치수선 기입
    for i in range(len(size_lines)):
        for j in range(len(size_lines[i])):
            if i == 0:
                pt1 = (size_lines[i][j][0] + 10, size_lines[i][j][1])
                pt2 = (size_lines[i][j][0] + 70, size_lines[i][j][1])
                cv2.line(result_image, pt1, pt2, (0, 0, 0), 1)
                cv2.line(sized_image, pt1, pt2, (0, 0, 0), 1)
                pt1 = (size_lines[i][j][2] + 10, size_lines[i][j][3])
                pt2 = (size_lines[i][j][2] + 70, size_lines[i][j][3])
                cv2.line(result_image, pt1, pt2, (0, 0, 0), 1)
                cv2.line(sized_image, pt1, pt2, (0, 0, 0), 1)
                pt1 = (size_lines[i][j][0] + 60, size_lines[i][j][1] - 10)
                pt2 = (size_lines[i][j][0] + 60, size_lines[i][j][3] + 10)
                cv2.line(result_image, pt1, pt2, (0, 0, 0), 1)
                cv2.line(sized_image, pt1, pt2, (0, 0, 0), 1)
                center = (size_lines[i][j][0] + 60, size_lines[i][j][1])
                cv2.circle(result_image, center, 2, (0, 0, 0), 2)
                cv2.circle(sized_image, center, 2, (0, 0, 0), 2)
                center = (size_lines[i][j][0] + 60, size_lines[i][j][3])
                cv2.circle(result_image, center, 2, (0, 0, 0), 2)
                cv2.circle(sized_image, center, 2, (0, 0, 0), 2)
            if i == 1:
```

### 첫 과정에서 얻은 치수 및 치수의 좌표 활용

1. 치수가 어느 방의 외벽과 가장 가까운지 계산
2. 각 치수가 할당된 도면의 선을 판단
3. 공간 바깥쪽에 일정한 간격으로 치수와 치수선을 입력

## 02 주요 코드



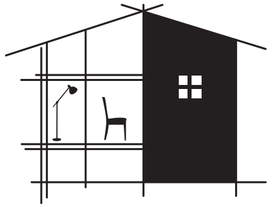
# 벽, 공간 기호 자동 배치

```
# 문 좌표 (좌상, 우상, 좌하, 우하)
    doors_points = [[], [], [], []]

    # 벽 내부의 실제 생활 공간 좌표
    inner_room = []

    for i in range(len(rooms_points)):
        left_boundary_check = True
        up_boundary_check = True
        right_boundary_check = True
        low_boundary_check = True

        pt1 = (rooms_points[i][0][0] + 3, rooms_points[i][0][1] + 3)
        pt2 = (rooms_points[i][1][0] - 3, rooms_points[i][1][1] - 3)
        if rooms_points[i][0][0] == upper_left[0]:
            pt1 = (rooms_points[i][0][0] + 10, pt1[1])
            left_boundary_check = False
        if rooms_points[i][0][1] == upper_left[1]:
            pt1 = (pt1[0], rooms_points[i][0][1] + 10)
            up_boundary_check = False
        if rooms_points[i][1][0] == lower_right[0]:
            pt2 = (rooms_points[i][1][0] - 10, pt2[1])
            right_boundary_check = False
        if rooms_points[i][1][1] == lower_right[1]:
            pt2 = (pt2[0], rooms_points[i][1][1] - 10)
            low_boundary_check = False

        cv2.rectangle(result_image, pt1, pt2, (255, 255, 255), -1)
        cv2.rectangle(result_image, pt1, pt2, (0, 0, 0), 2)
        inner_room.append([pt1, pt2])

        if left_boundary_check and up_boundary_check:
            doors_points[0].append([(pt1[0] + 5, pt1[1]), (pt1[0] + 50, pt1[1] - 6)])
        elif right_boundary_check and up_boundary_check:
            doors_points[1].append([(pt2[0] - 5, pt1[1]), (pt2[0] - 50, pt1[1] - 6)])
        elif left_boundary_check and low_boundary_check:
            doors_points[2].append([(pt1[0], pt2[1] - 5), (pt1[0] - 6, pt2[1] - 50)])
        elif right_boundary_check and low_boundary_check:
            doors_points[3].append([(pt2[0], pt2[1] - 5), (pt2[0] + 6, pt2[1] - 50)])
```

### 외벽, 내벽을 판별하여 수행

1. 도면 끝 좌표 네 개를 기준으로 삼음
2. 각 벽면이 전체를 감싸는 외벽인지, 공간을 구분하는 내벽인지 판단하여 벽 기호를 작성
   - 2.1. 외벽의 경우, 내벽의 두 배 두께로 묘사
3. 각 공간마다 한 개의 문을 배치
   - 3.1. 해당 공간을 감싸는 벽이 내벽일 경우 문이 필요하다 판단
   - 3.2. 해당 공간 좌표에 문 기호 배치

## 02 주요 코드



# 실내 인테리어 배치

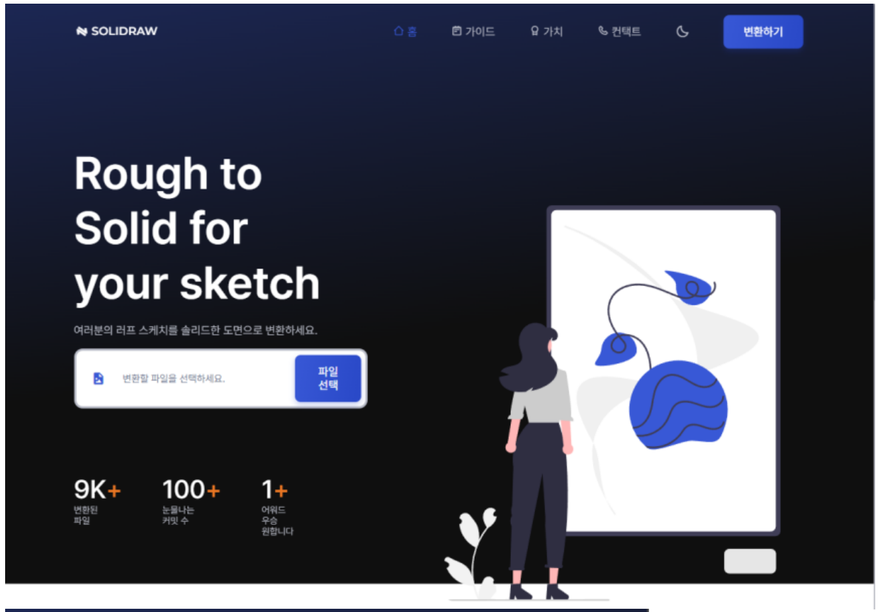
```
        for j in range(len(OCR_room)):
            if inner_room[i][0][0] < OCR_room[j][1] < inner_room[i][1][0]:
                if inner_room[i][0][1] < OCR_room[j][2] < inner_room[i][1][1]:
                    if OCR_room[j][0] == '화장실':
                        # print(OCR_room[j][0])
                        # print(inner_room[i])
                        # print(left_boundary_check, up_boundary_check, right_boundary_check, low_boundary_check)

                        if left_boundary_check and up_boundary_check:
                            pt1 = (inner_room[i][0][0], inner_room[i][0][1] + 5)
                            pt2 = (inner_room[i][1][0], inner_room[i][0][1] + 5)
                            cv2.line(result_image, pt1, pt2, (0, 0, 0), 1)

                            pt1 = (inner_room[i][0][0] + 10, inner_room[i][0][1] + 25)
                            pt2 = (inner_room[i][0][0] + 45, inner_room[i][0][1] + 5)
                            cv2.rectangle(result_image, pt1, pt2, (0, 0, 0), 1)
...
 if OCR_room[j][0] == '주방':
                        # print(OCR_room[j][0])
                        # print(inner_room[i])
                        # print(left_boundary_check, up_boundary_check, right_boundary_check, low_boundary_check)

                        if left_boundary_check and up_boundary_check:
                            pt1 = (inner_room[i][0][0], inner_room[i][0][1] + 30)
                            pt2 = (inner_room[i][1][0], inner_room[i][0][1] + 30)
                            cv2.line(result_image, pt1, pt2, (0, 0, 0), 1)

                            pt1 = (inner_room[i][0][0] + 15, inner_room[i][0][1] + 25)
                            pt2 = (inner_room[i][0][0] + 50, inner_room[i][0][1] + 5)
                            cv2.rectangle(result_image, pt1, pt2, (0, 0, 0), 1)

                            pt1 = (inner_room[i][0][0] + 17, inner_room[i][0][1] + 23)
                            pt2 = (inner_room[i][0][0] + 48, inner_room[i][0][1] + 7)
                            cv2.rectangle(result_image, pt1, pt2, (0, 0, 0), 1)

                            pt1 = (inner_room[i][0][0] + 33, inner_room[i][0][1] + 5)
                            pt2 = (inner_room[i][0][0] + 33, inner_room[i][0][1] + 15)
```

### 각 공간 정보에 부합하는 실내 인테리어 배치

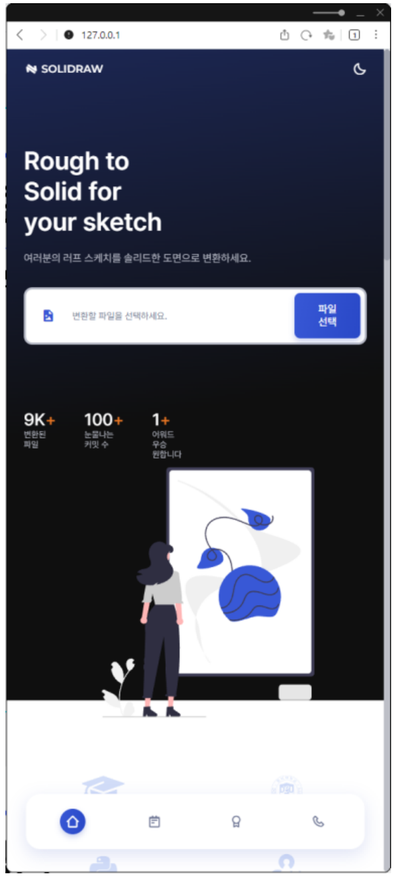
1. 첫 번째 과정에서 읽은 공간 메모와 해당 메모의 좌표를 불러옴
2. 해당 메모가 어느 방의 중심점과 가장 가까운지 계산
3. 각 방의 용도 판단
4. 사전에 설정된 각 공간별 고정 실내 인테리어(예. 주방은 싱크대, 화장실은 세면대) 를 공간 내부 규격에 맞게 배치

03 GUI

# 웹앱을 활용한 편리한 사용

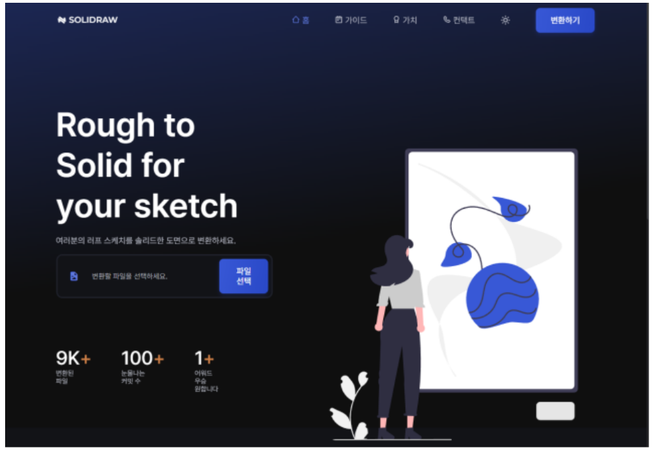



사용자에게 편리한 UI 제공




PC, 태블릿,
모바일 지원!

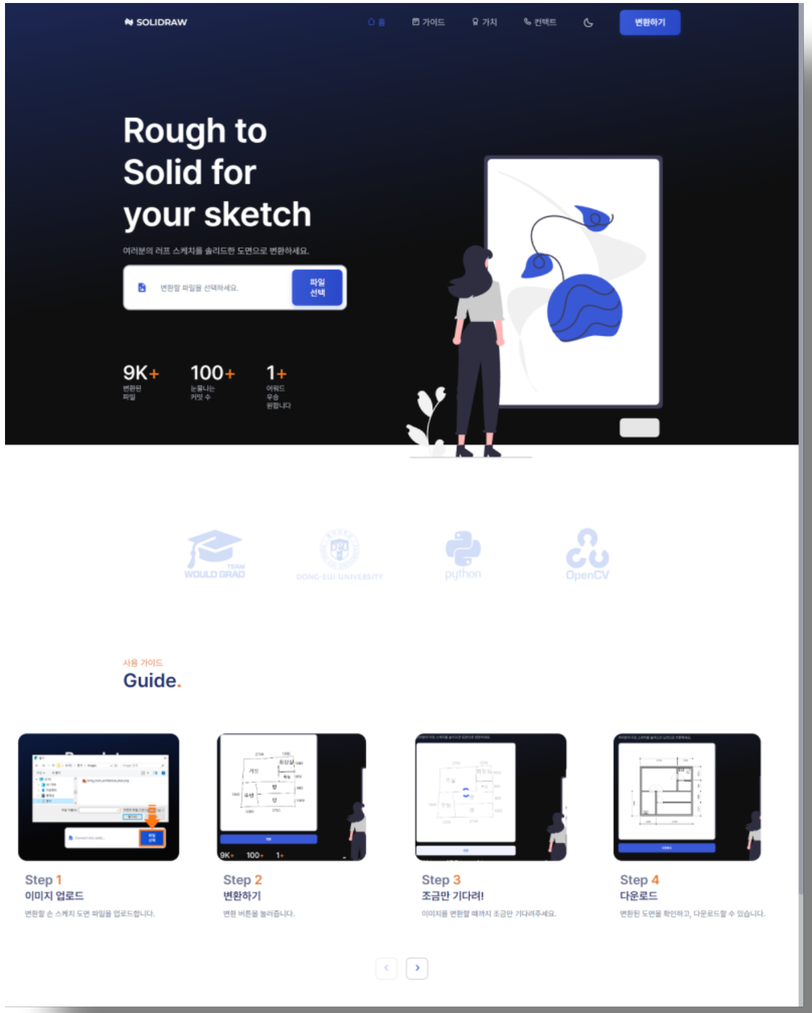

사용 가이드 확인 가능!

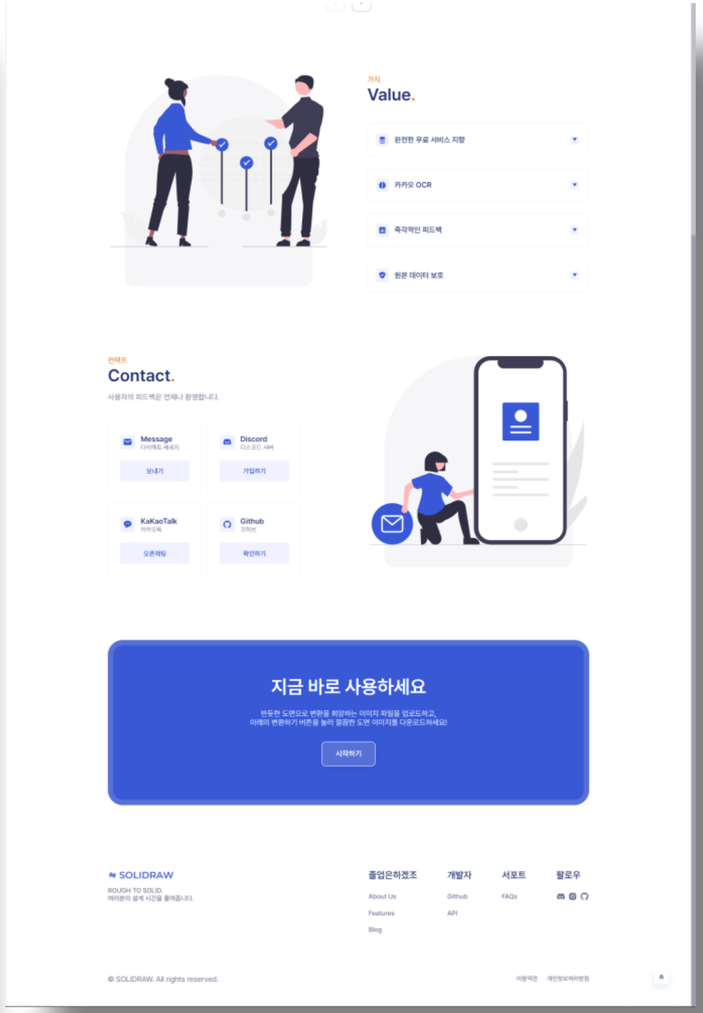



다크모드 지원!

# 03 GUI

# 웹 전체 화면


SOLIDRAW
변환하기
Rough to
Solid for
your sketch
파일 선택
9K+
100+
1+
사용 가이드
Guide.
Step 1
이미지 업로드
Step 2
변환하기
Step 3
조금만 기다려!
Step 4
다운로드


Value.
Contact.
Message
Discord
KaKaoTalk
Github
지금 바로 사용하세요
시작하기
SOLIDRAW
개발자
서포트
팔로우

## 04 진행

# Git과 Github를 활용한 버전관리

### Push 및 Merge 실행 내역

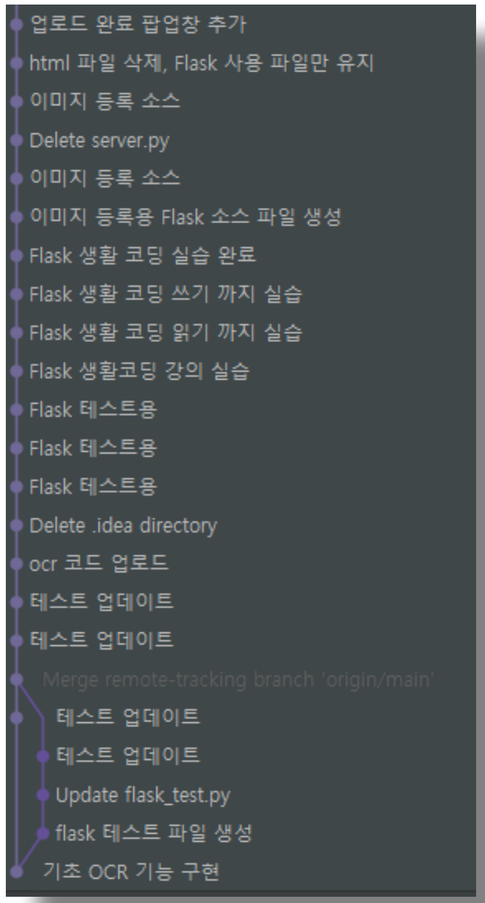
| | | | | |
|---|---|---|---|---|
| Delete .idea directory<br>dyson1357 • Jun 2, 2022 | read me 업데이트!<br>dyson1357 • 2 days ago | Update README.md<br>dyson1357 • 2 days ago | 손 도면 변환 프로그램<br>kkamag • 2 days ago | 오타 수정<br>dyson1357 • 1 hour ago |
| ocr 코드 업로드<br>dyson1357 • Jun 2, 2022 | 수정<br>dyson1357 • 3 days ago | 이미지 등록, 확인, 변환<br>dyson1357 • 2 days ago | Update upload_img.py<br>dyson1357 • 2 days ago | 완성...<br>khakhiD • 6 hours ago |
| 테스트 업데이트<br>dyson1357 • Jun 2, 2022 | 수정<br>dyson1357 • 3 days ago | 이미지 표시 캐시 문제 해결<br>dyson1357 • 2 days ago | 주석 작업 중<br>dyson1357 • 2 days ago | 완성...<br>khakhiD • 6 hours ago |
| 테스트 업데이트<br>dyson1357 • Jun 2, 2022 | Delete KakaoTalk_20220607_1209579...<br>dyson1357 • 3 days ago | Add files via upload<br>kkamag • 2 days ago | 버튼들 추가<br>dyson1357 • 2 days ago | 병합충돌 해결<br>khakhiD • 11 hours ago |
| Merge remote-tracking branch 'origi...<br>dyson1357 • Jun 2, 2022 | 업로드된 이미지 파일<br>dyson1357 • 3 days ago | 이미지 이름 변환한 뒤 업로드 및 저장<br>dyson1357 • 2 days ago | 이미지 경로 수정<br>dyson1357 • 2 days ago | Merge branch 'main' of https://githu...<br>khakhiD • 11 hours ago |
| 테스트 업데이트<br>dyson1357 • Jun 2, 2022 | 이미지 드롭다운으로 추가<br>dyson1357 • 3 days ago | 이미지 경로 재 지정<br>dyson1357 • 2 days ago | Delete ocr.py<br>dyson1357 • 2 days ago | 수정중<br>khakhiD • 11 hours ago |
| 테스트 업데이트<br>dyson1357 • Jun 2, 2022 | 업로드한 파일 목록 출력<br>dyson1357 • 3 days ago | 이미지 처리를 위한 img, css, js 파일 ...<br>dyson1357 • 2 days ago | 입력 이미지<br>kkamag • 2 days ago | 복구<br>dyson1357 • 17 hours ago |
| Update flask_test.py<br>dyson1357 • Jun 2, 2022 | 이미지 드롭다운으로 추가<br>dyson1357 • 3 days ago | Delete static/img directory<br>dyson1357 • 2 days ago | 출력 이미지<br>kkamag • 2 days ago | image source edit<br>khakhiD • 18 hours ago |
| flask 테스트 파일 생성<br>dyson1357 • Jun 2, 2022 | 안댐<br>dyson1357 • 3 days ago | Add files via upload<br>dyson1357 • 2 days ago | 손 도면 변환 프로그램<br>kkamag • 2 days ago | 출력 이미지<br>kkamag • 2 days ago |
| 기초 OCR 기능 구현<br>dyson1357 • May 29, 2022 | 업로드한 파일 목록 출력<br>dyson1357 • 3 days ago | 등록된 사진 확인 가능하게 수정<br>dyson1357 • 2 days ago | Delete test_img.png<br>kkamag • 2 days ago | 입력 이미지<br>kkamag • 2 days ago |

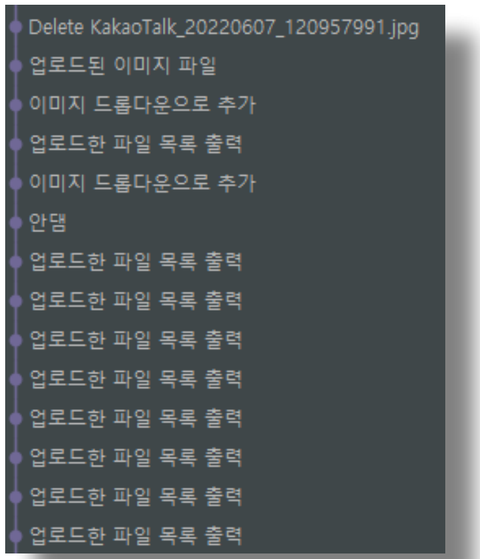
** 사소한 업데이트 사항은 제외하고 첨부하였음*

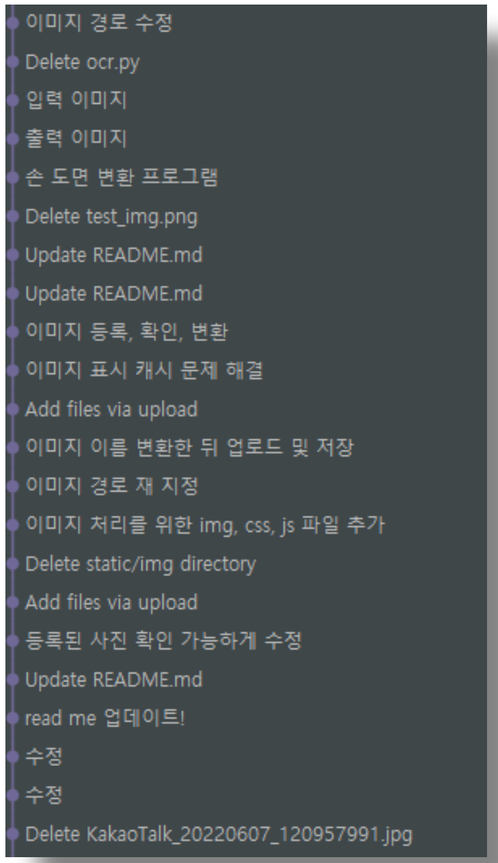
## 04 진행

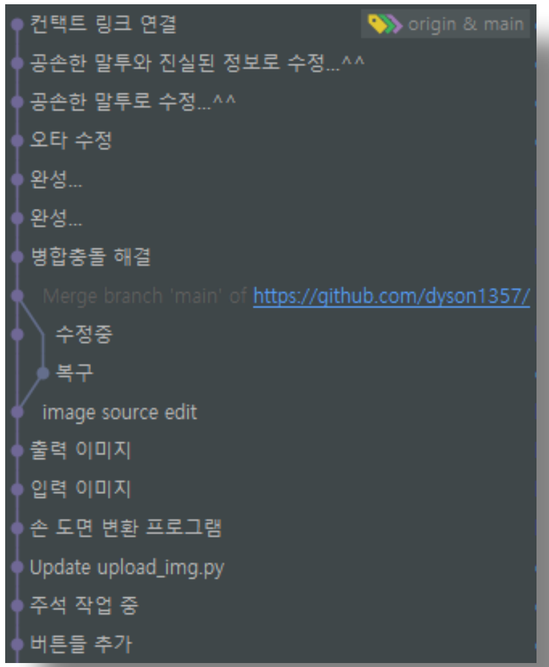
# Git과 Github를 활용한 버전관리

### 전체 Branch 그림




*＊사소한 업데이트 사항은 제외하고 첨부하였음*

# 03

## 역할 및 소감

01. 역할 및 소감

# 03 역할 및 소감

# 역할과 소감

### 김도영 | 팀장 | 프로젝트 관리, 핵심 변환 알고리즘 구현

초기 계획보다 구현되지 못한 기능이 많아서 아쉬웠으나, 기반이 되는 기능은 완성하여 프로젝트에 대한 가능성을 확인할 수 있는 작업이었다. 성능 향상을 위한 분석에도 많은 도움이 되었으며, 캡스톤 강의가 끝난 이후에 더 뛰어난 성능의 프로그램으로 개선시키고자 합니다.

### 신동호 | 팀원 | 디자인, 웹 프론트엔드 구현, 서버 연동, 버전관리

재학 중 학습한 내용을 토대로 팀원들과 개발 역할군을 나누어 작업했던 부분에서 개인적인 발전이 있었기에 즐거웠습니다. 프로젝트를 진행하면서 기존에 배워보고자 했던 프레임워크를 익혀가는 과정에서 다양한 프로젝트 아이디어가 생겼습니다. 이번 프로젝트를 보완하면서 더욱 기술을 익혀 더욱 완성도 높은 프로젝트를 진행해보고자 합니다.

### 손다연 | 팀원 | 건축과 자문, OCR 구현, 서버 구현, 버전관리

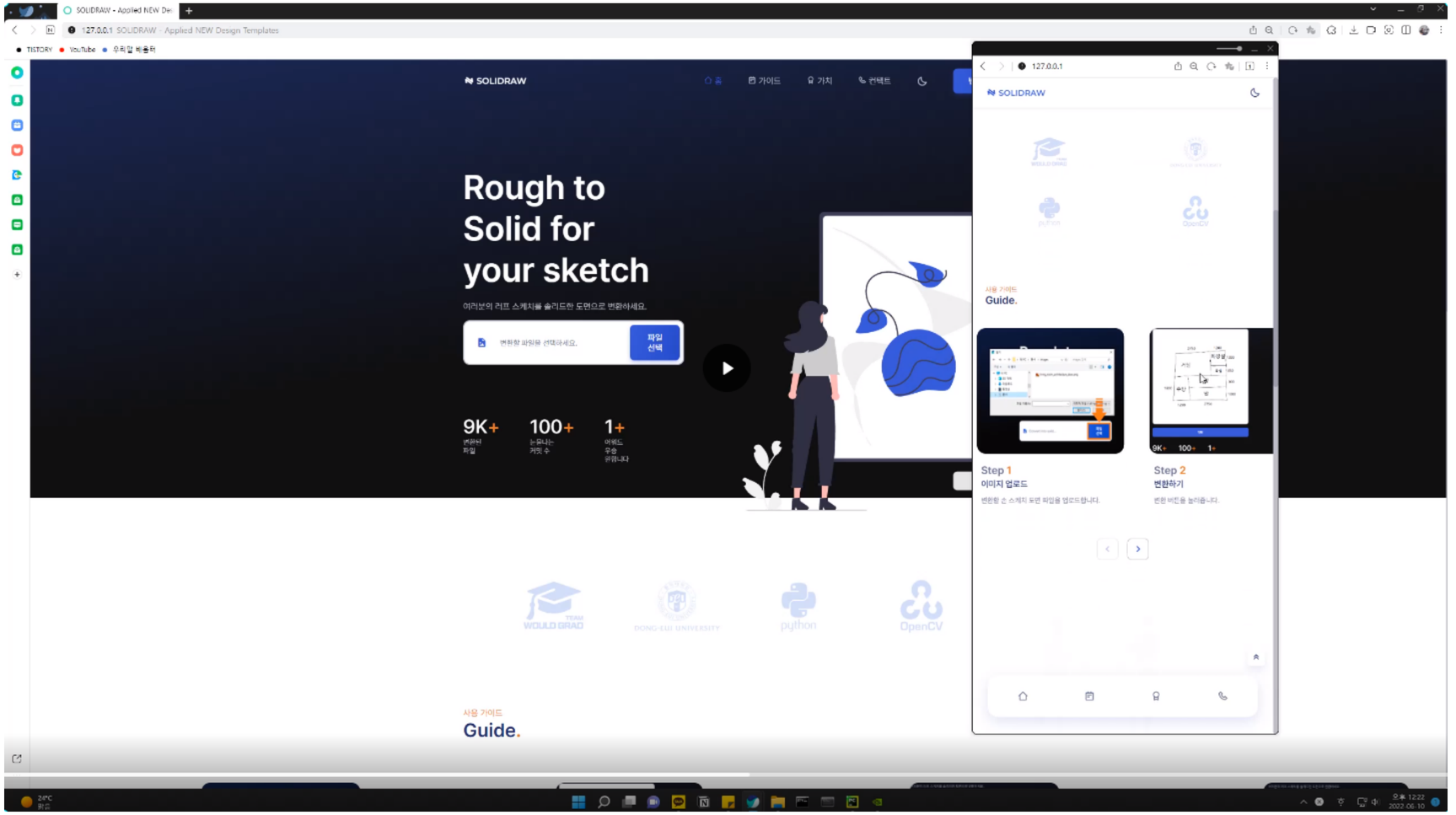
그동안 교재 실습 수준의 프로젝트를 해 오다가 새로운 주제를 고안하고, 조사하고, 구현하는 과정이 재미있었습니다. 특히 그동안 배웠던 내용들을 종합하여 구현해보는 과정이 캡스톤이라는 과목명을 잘 담은 것 같다는 생각이 들면서도 뿌듯했습니다. 또, 버전관리를 통해 제대로된 프로젝트를 해낸 것 같다 느꼈으며 애정이 있는 프로젝트인 만큼 더 발전시켜 좋은 포트폴리오로 활용하고 싶습니다.

# 04

## 시연

01. 시연 영상

# 04 시연 영상


SOLIDRAW
Rough to
Solid for
your sketch
9K+
100+
1+
Guide.
Step 1
이미지 업로드
Step 2
변환하기

SOLIDRAW - Applied NEW Des

127.0.0.1:5000/#contact

TISTORY YouTube 우리말 배움터

SOLIDRAW

홈 가이드 가치 컨택트

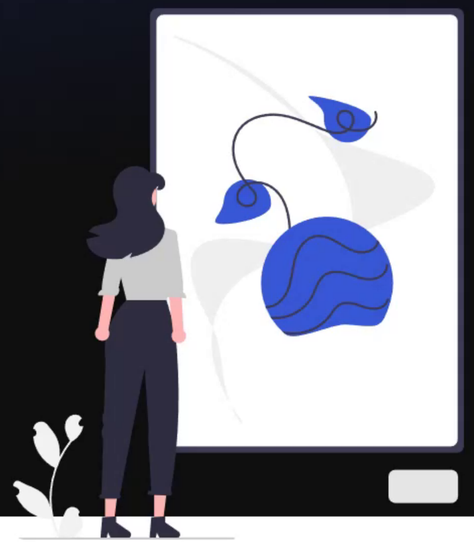
변환하기

# Rough to Solid for your sketch

여러분의 러프 스케치를 솔리드한 도면으로 변환하세요.

변환할 파일을 선택하세요.

파일 선택



9K+
변환된 파일

100+
눈물나는 커밋 수

1+
어워드 우승 원합니다

TEAM WOULD GRAD

DONG-EUI UNIVERSITY

python

OpenCV

사용 가이드

## Guide.

24°C 맑음

오후 12:21 2022-06-10

# 감사합니다