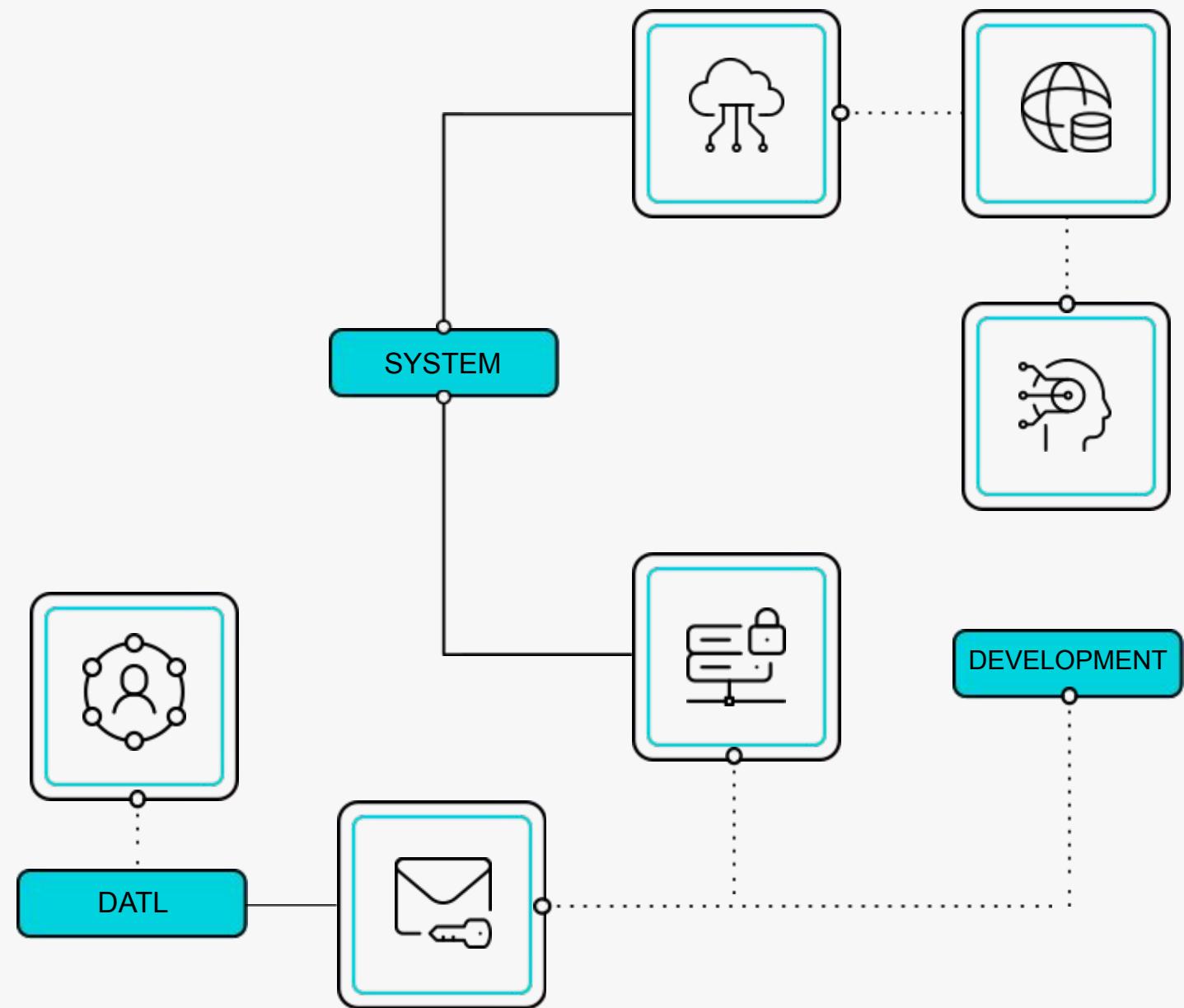


숫자 맞추기

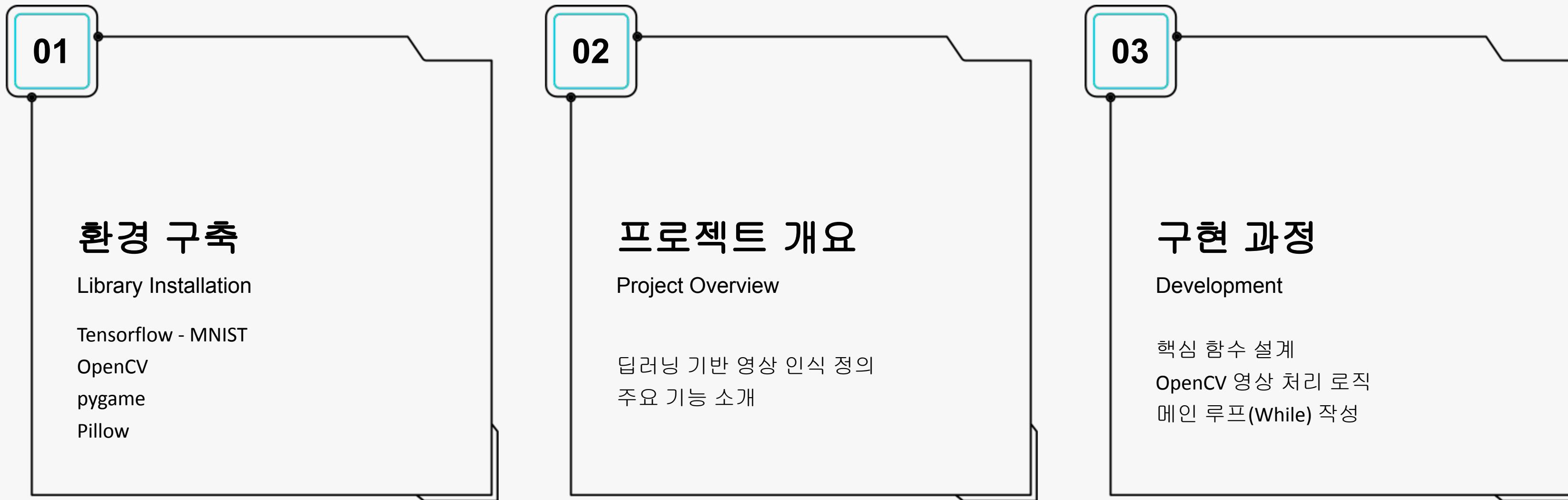
2026.01 머신러닝 / Tensorflow & OpenCV

신재훈



Contents

목차



01

환경 구축

Chapter 01



본 프로젝트 구현을 위해 필요한 개발 환경 및 라이브러리 설치 과정을 소개합니다.

Tensorflow, OpenCV, Pygame, Pillow

01

환경 구축

Tensorflow



TensorFlow

- 구글에서 만든 딥러닝 프로그램을 쉽게 구현할 수 있도록 기능을 제공하는 라이브러리
- 기본적으로 C++로 구현되나, 파이썬 라이브러리 사용을 권한다.

OpenCV



OpenCV

- Open Source Computer Vision Library
- 실시간 영상 및 이미지 처리에 특화된 강력한 오픈 소스 라이브러리
- 실시간 처리, 객체 탐지/추적, 얼굴인식, 이미지 필터링 등

01

환경 구축

Pillow



pillow

- PIL(Python Imaging Library)의 후속 프로젝트
- 파이썬에서 이미지를 열기, 수정, 저장 등 다양한 이미지 처리 기능을 제공하는 핵심 라이브러리

Pygame



- 파이썬 언어로 2D 게임 및 멀티미디어 프로그램을 만들기 위해 설계된 오픈 소스 라이브러리
- 크로스 플랫폼, 간편한 멀티미디어 처리, 무료 오픈소스, 쉬운 학습

01

환경 구축

Pillow

```
import sys
from logging import exception

from fontTools.misc.psOperators import ps_string

# pip 설치 시도
subprocess.check_call([sys.executable, "-m", "ensurepip", "--default-pip"])
# Pillow 설치
subprocess.check_call([sys.executable, "-m", "pip", "install", "Pillow"])
✓ [2] 5초 218밀리초
```

!pip install Pillow

Unicode Support

파이썬의 한글 깨짐(폰트 오류) 문제 해결

Font(.ttf)

외부 폰트(.ttf) 데이터를 활용한 한글
텍스트 렌더링

01

환경 구축

pygame

```
import pygame
import os

pygame.mixer.init()

# 윈도우 기본 미디어 폴더 경로
win_media_path = "C:/Windows/Media/"

sound_correct = pygame.mixer.Sound(win_media_path + "Speech On.wav")
sound_wrong = pygame.mixer.Sound(win_media_path + "Speech Misrecognition.wav")
```

!pip install pygame

mixer

pygame 라이브러리에서 사운드 재생을
담당

Sound

윈도우 기본 미디어 파일 사용하여 재생

02

프로젝트 개요

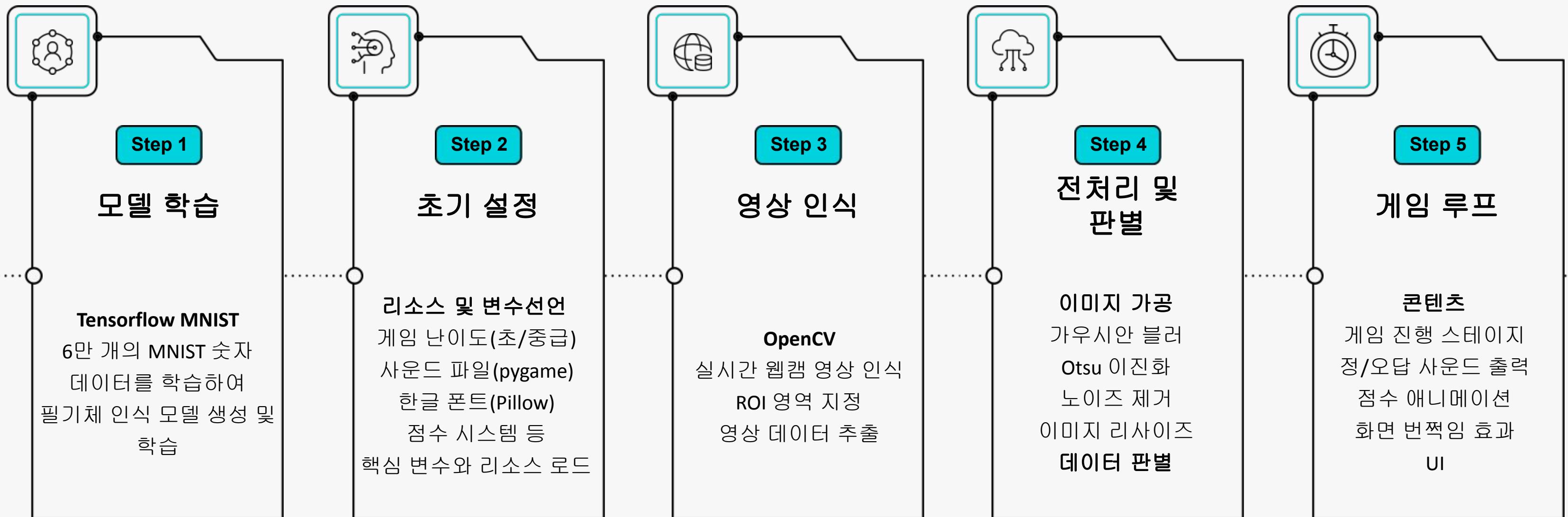
Chapter 02



본 프로젝트는 실시간 영상 인식을 통한 숫자 맞추기 게임으로, 텐서플로를 활용한 모델 학습부터 데이터 전처리, 그리고 사용자 인터페이스 구현까지의 과정을 포함하고 있습니다.

02

프로젝트 개요



03

구현 과정

Chapter 03



본 프로젝트의 핵심 기능을 구현하기 위한 단계별 소스코드 구현 상세 내용을 다루고 있습니다..

03-1 구현 과정 - 함수 및 UI/UX

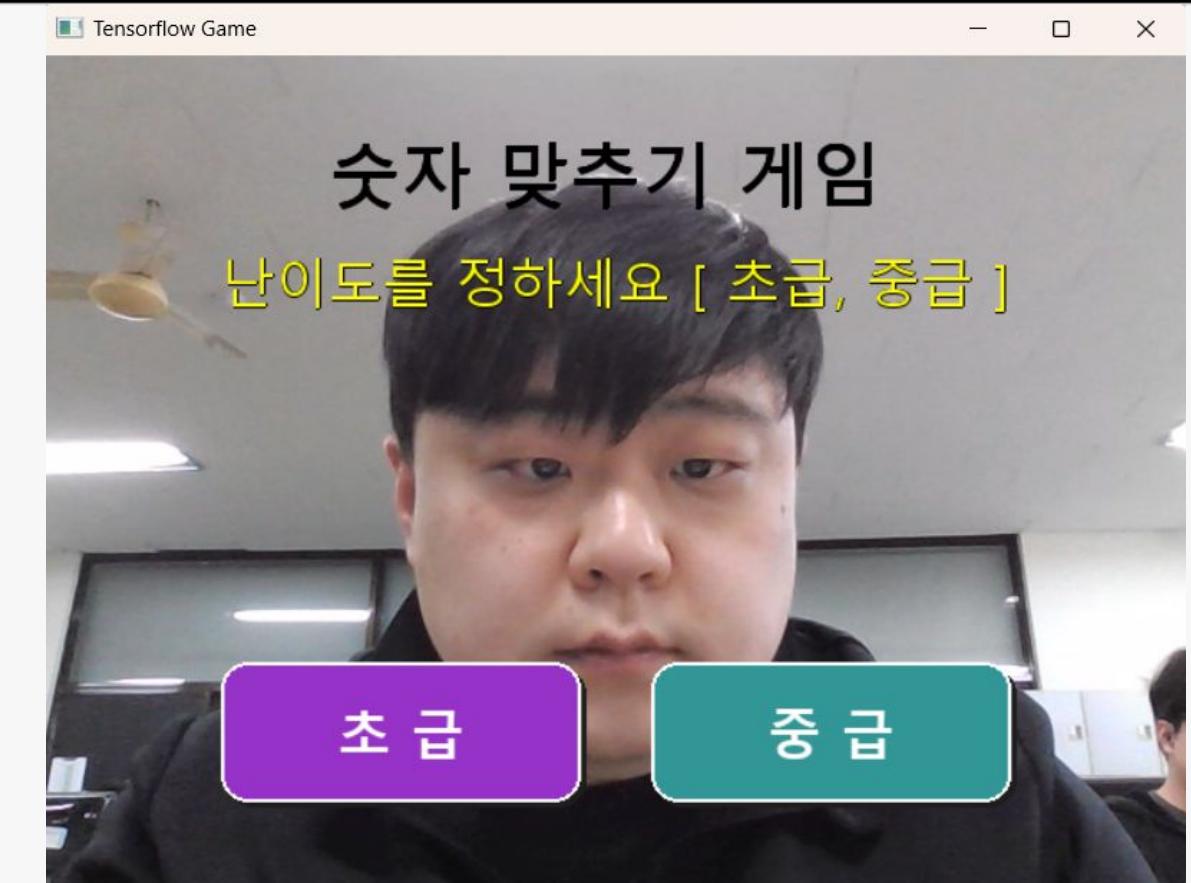
print_korean()

```
# 한글 출력을 위해서는 opencv2 프레임을 Pillow로 전환 후, 한글로 그린 뒤에 다시 opencv2에 돌려주는 함수가 필요하다.

def print_korean(img, text, pos, f_size, _color):
    img_pil = Image.fromarray(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    draw = ImageDraw.Draw(img_pil)
    try:
        _font = ImageFont.truetype("C:/Windows/Fonts/malgun.ttf", f_size)
    except Exception as e:
        _font = ImageFont.load_default()
        print(f"error : {e}")

    # 글씨 외곽선 그리기
    x, y = pos
    stroke_color = (0, 0, 0) # 검은색 외곽선
    for adj in range(0, 2): # 두께 조절 (2픽셀)
        draw.text((x+adj, y), text, font=_font, fill=stroke_color)
        draw.text((x, y+adj), text, font=_font, fill=stroke_color)

    draw.text(pos, text, font=_font, fill=_color)
    return cv2.cvtColor(np.array(img_pil), cv2.COLOR_RGB2BGR)
```



- Pillow 라이브러리를 사용하여 한글 깨짐 해결

03-1 구현 과정 - 함수 및 UI/UX

draw_button()

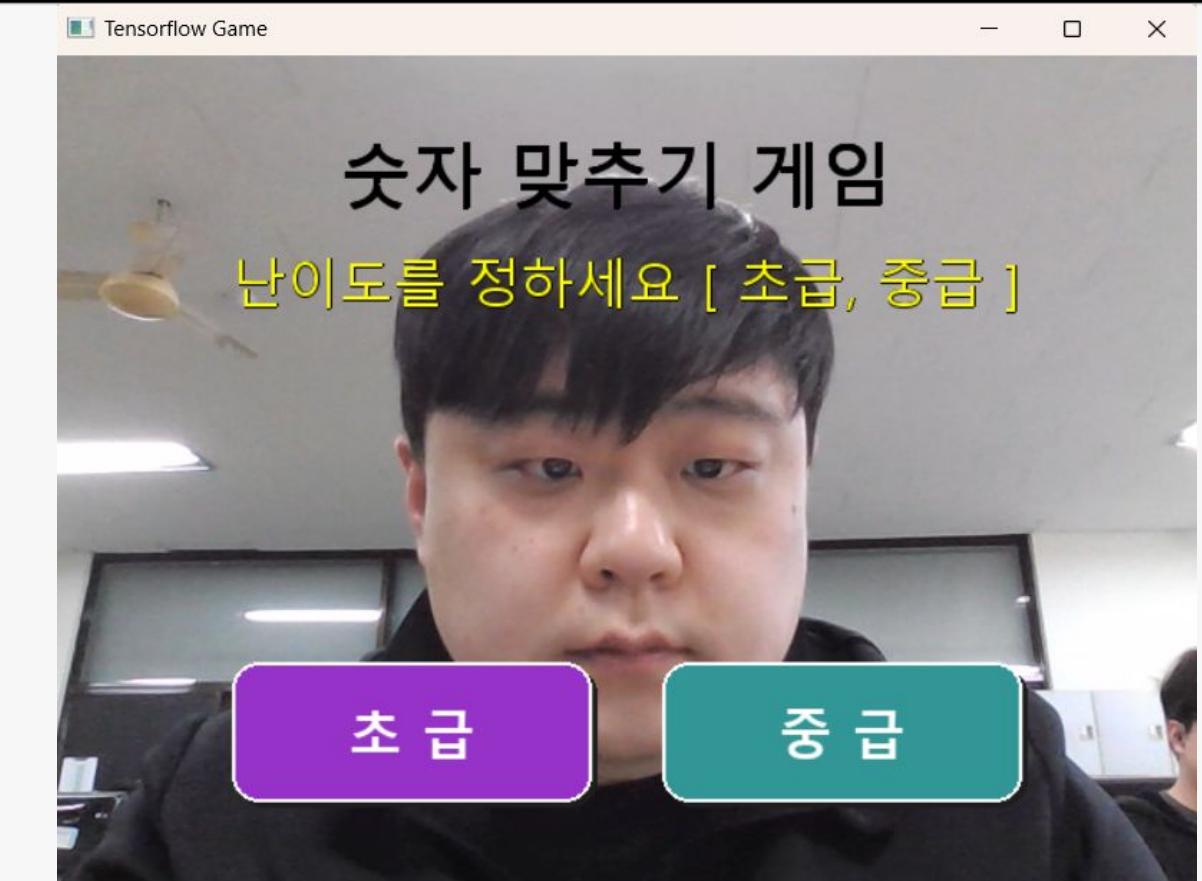
```
def ui_작용하기
    draw_button(img, text, rect, base_color, text_color =(255,255,255)):
        x1, y1, x2, y2 = rect
        img_pil = Image.fromarray(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        draw = ImageDraw.Draw(img_pil)
        try:
            _font = ImageFont.truetype("C:/Windows/Fonts/malgunbd.ttf", 30)
        except:
            _font = ImageFont.load_default()

        # 1. 외곽선이 있는 둥근 사각형 (버튼 몸체)
        draw.rounded_rectangle([x1+3, y1+3, x2+3, y2+3], radius=15, fill=(30, 30, 30)) # 그림자
        draw.rounded_rectangle([x1, y1, x2, y2], radius=15, fill=base_color, outline=(255, 255, 255), width=2)

        # 2. 텍스트 중앙 정렬 계산
        w, h = x2 - x1, y2 - y1
        tw, th = draw.textsize(text, font=_font) if hasattr(draw, 'textsize') else (70, 30)
        text_pos = (x1 + (w - tw) / 2, y1 + (h - th) / 2 - 5)

        # 3. 버튼 텍스트 그리기
        draw.text(text_pos, text, font=_font, fill=(255, 255, 255)) # 버튼 글씨는 흰색 고정

    return cv2.cvtColor(np.array(img_pil), cv2.COLOR_RGB2BGR)
```



- UI/UX를 위한 초/중급 버튼
- 외곽선이 있는 둥근 모서리 사각형

03-1 구현 과정 - 함수 및 UI/UX

mouse_callback()

```
### 난이도 선택을 위한 마우스 클릭  
def mouse_callback(event, _x, _y, _flags, _param):  
    global game_mode, game_level, start_time, current_mouse_pos  
  
    # 마우스가 움직일 때마다 좌표 업데이트  
    if event == cv2.EVENT_MOUSEMOVE:  
        current_mouse_pos = [_x, _y]  
  
    if event == cv2.EVENT_LBUTTONDOWN and game_mode == "WAITING":  
        if (btn1_rect[0] < _x < btn1_rect[2]) and (btn1_rect[1] < _y < btn1_rect[3]):  
            game_level = 1  
            game_init()  
            game_mode = "READY"  
            print("초급")  
        elif (btn2_rect[0] < _x < btn2_rect[2]) and (btn2_rect[1] < _y < btn2_rect[3]):  
            game_level = 2  
            game_init()  
            game_mode = "READY"  
            print("중급")  
        else: # 난이도 고급  
            pass
```

quiz_fuc()

```
### 퀴즈 메소드  
def quiz_fuc(lv):  
    global answer, quiz_text  
    if lv == 1: # 초급  
        answer = random.randint(1, 9) # 1~9까지 숫자 랜덤  
        #answer = 4 # test 용  
        quiz_text = "숫자 : " + str(answer)  
        print(quiz_text)  
        pass  
    elif lv == 2: # 중급  
        op = random.choice(['+', '-'])  
        if op == '+':  
            a = random.randint(1, 8)  
            b = random.randint(1, 9-a) # 합이 9가 넘지 않도록, a 기반으로 랜덤  
            answer = a + b  
        else:  
            a = random.randint(2, 9)  
            b = random.randint(1, a-1) # 합이 음수가 되지않을고 a보다 작은 숫자만 나오도록  
            answer = a - b  
        quiz_text = f"QUIZ : {a} {op} {b} = ?" # ex) 5 + 4 = ? / 9 - 4 = ?  
        print(quiz_text)  
    elif lv == 3: # 고급  
        print("여기 아이디어의 단계입니다.")
```

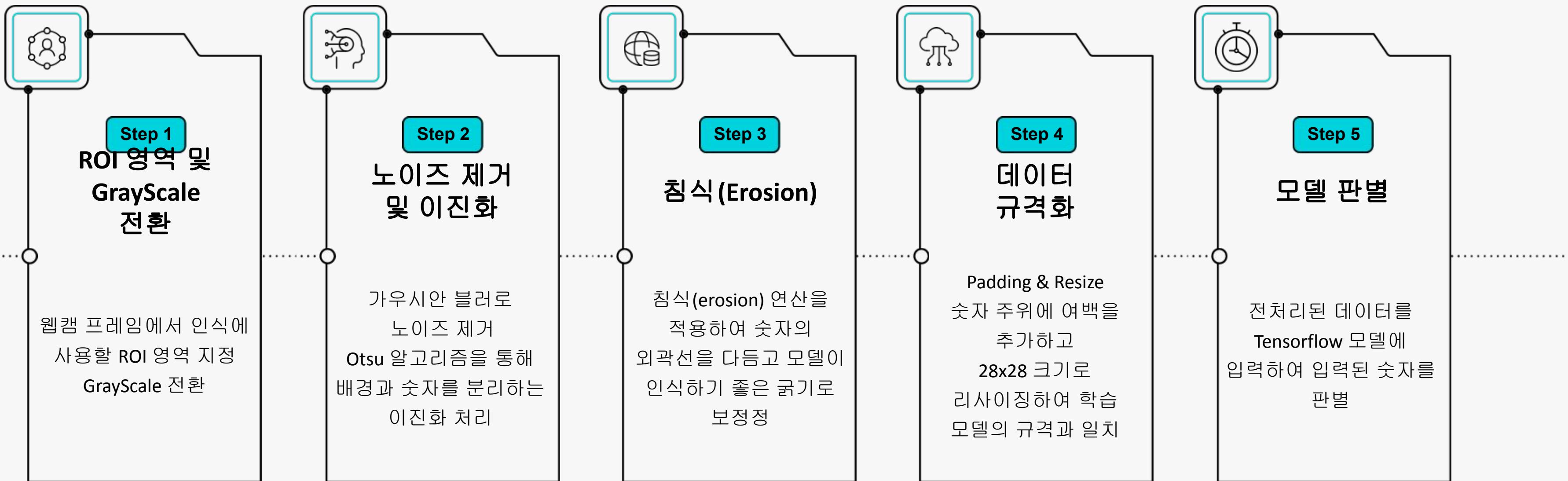
03-1 구현 과정 - 함수 및 UI/UX

```
def game_init():
    global quiz_count, total_score, start_time
    quiz_count = 1
    total_score = 0.0
    quiz_fuc(game_level)
    start_time = time.time()
```

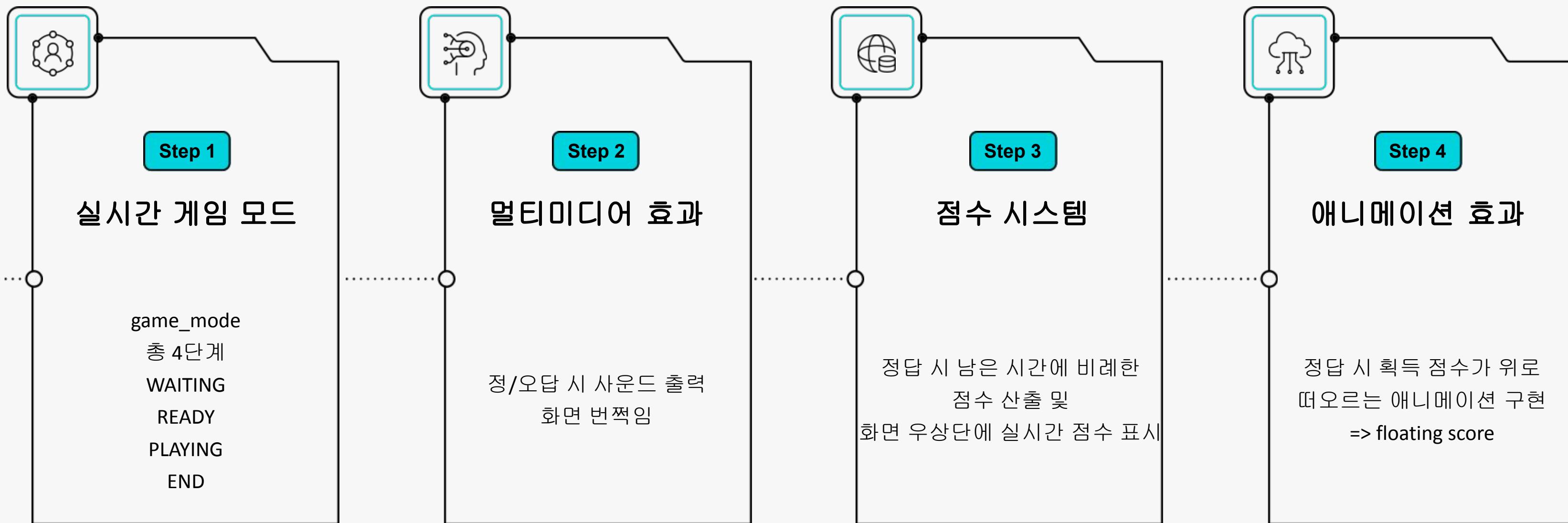
- 01** `mouse_callback()`
실시간 마우스 이벤트 처리
사용자가 화면 상의 버튼과 상호작용할 수 있도록 합니다.
- 02** `quiz_fuc()`
게임 난이도에 따른 로직
선택된 난이도에 따라 단일숫자 또는 사칙연산 퀴즈를
랜덤으로 생성하여 게임을 진행하도록 합니다.
- 03** `game_init()`
게임 초기화
새로운 게임 시작 시 점수, 퀴즈 횟수, 시작 시간 등 초기값으로
재설정하여 문제없이 진행할 수 있도록 합니다.

03-2

구현 과정 - 영상 전처리 및 데이터 판별



03-3 구현 과정 - 메인 루프



03-3 구현 과정 - 메인 루프 - WAITING

```
###  
while True:  
    ret, frame = cap.read() # 리턴값과 프레임  
    if not ret: # false면  
        print("프레임을 가져 올 수 없습니다.")  
        break  
    key = cv2.waitKey(1) & 0xFF  
  
    flip_frame = cv2.flip(frame, 1) # y축을 기준으로 뒤집어 주세요 -> 거울 반전  
    height, width, color = frame.shape # 높이, 폭, color인데 color는 딱히 안쓸거임  
    center_x, center_y = width // 2, height // 2 # x와 y의 중심점  
  
    offset_y = 60 # 화면 위아래 조절하려고 만든 변수  
    roi = flip_frame[center_y - 150 + offset_y : center_y + 150 + offset_y, center_x - 150:center_x + 150] # roi 영역 만들기  
    # #x,y중심점을 기준으로 150씩의 거리를 가진 영역 만들기 -> 300x300  
  
    font = cv2.FONT_HERSHEY_SIMPLEX
```

03-3 구현 과정 - 메인 루프 - WAITING

```
# game_mode 구현.  
  
if game_mode == "WAITING": # 난이도 선택  
    flip_frame = print_korean(flip_frame, "숫자 맞추기 게임", (center_x-160, center_y-200), 40, (0,0,0))  
    flip_frame = print_korean(flip_frame, "난이도를 정하세요 [ 초급, 중급 ]", (center_x-220, center_y-130), 30, (255,255,0))  
  
    mx, my = current_mouse_pos # 현재 마우스 좌표  
  
    # 초급 버튼 호버과 - 더 밝은 보라색  
    if (btn1_rect[0] < mx < btn1_rect[2]) and (btn1_rect[1] < my < btn1_rect[3]):  
        flip_frame = draw_button(flip_frame, "초 급", btn1_rect, (200, 100, 255))  
    else:  
        flip_frame = draw_button(flip_frame, "초 급", btn1_rect, (150, 50, 200))  
  
    # 중급 버튼 호버과 - 더 밝은 청록색  
    if (btn2_rect[0] < mx < btn2_rect[2]) and (btn2_rect[1] < my < btn2_rect[3]):  
        flip_frame = draw_button(flip_frame, "중 급", btn2_rect, (100, 220, 220))  
    else:  
        flip_frame = draw_button(flip_frame, "중 급", btn2_rect, (50, 150, 150))
```

01

상단 UI 출력

print_korean() 함수를 사용하여 화면 상단에 게임 타이틀과 난이도 안내 문구를 출력합니다.

02

버튼 호버
효과

현재 마우스 좌표를 받아, 버튼 영역(btn_rect)내에 있는지 판별하여, 마우스가 올라갔을 때 호버 효과를 내도록 합니다.

03

색상
업데이트

if-else문을 통해 호버 효과를 낼 때, 조건에 따라 다른 RGB 값으로 변경되도록 합니다.

03-3 구현 과정 - 메인 루프 - READY

```
elif game_mode == "READY":  
    flip_frame = print_korean(flip_frame, "게임이 곧 시작됩니다!!!", (center_x-215, center_y-120), 40, (255,0,0))  
  
    delay_time = time.time() - start_time  
    count = int(ready_time - delay_time) + 1  
  
    if count > 0:  
        flip_frame = print_korean(flip_frame, str(count), (center_x - 15, center_y - 25), 50, (255,0,0))  
    else:  
        start_time = time.time()  
        game_mode = "PLAYING"
```

01

안내 문구

print_korean() 함수를 사용하여 “게임이 곧 시작됩니다!!!”라는 메시지를 화면 중앙에 배치하여 사용자에게 안내합니다.

02

카운트

time.time()을 활용해 설정된 준비 시간(ready_time)으로부터 남은 시간을 산출합니다.
이 때, 화면에 카운트는 정수(in)로 표시합니다..

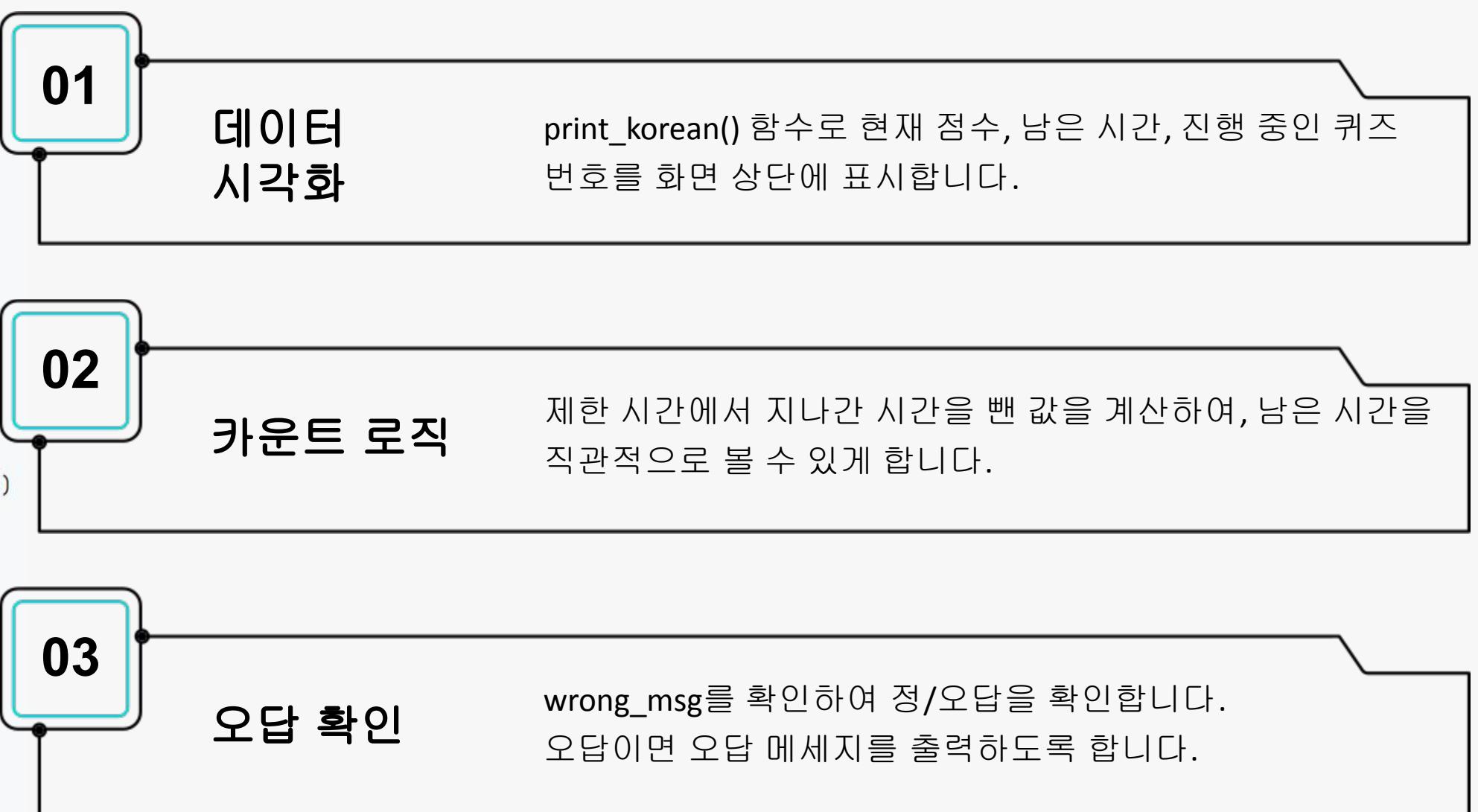
03

게임 모드
전환

카운트 다운이 0이 되면, game_mode를 PLAYING으로 전환하여, 실제로 게임을 진행하는 PLAYING 모드로 넘어갑니다.
이 때, start_time을 재설정하여 문제가 없도록 합니다.

03-3 구현 과정 - 메인 루프 - PLAYING - 1/5

```
elif game_mode == "PLAYING":  
    cv2.rectangle(flip_frame, (center_x - 150, center_y - 150 + offset_y),  
                 (center_x + 150, center_y + 150 + offset_y), (0,0,255), 2) # 외곽선 그리기  
  
    delay_time = time.time() - start_time  
    remain_time = max(0, time_limit - delay_time)  
  
    # 상단 정보 표시  
    flip_frame = print_korean(flip_frame, quiz_text, (0, 60), 40, (255, 255, 0))  
    flip_frame = print_korean(flip_frame, f"남은 시간: {remain_time:.1f}초", (0, 110), 30, (0, 0, 255))  
    # 2. 진행 상황 표시 (중상단)  
    flip_frame = print_korean(flip_frame, f"{quiz_count} / {max_quiz}", (center_x - 50, 20), 35, (255, 255, 255))  
    # 3. 실시간 점수 표시 (우상단)  
    flip_frame = print_korean(flip_frame, f"SCORE: {total_score:.1f}", (width - 220, 20), 35, (0, 255, 0))  
  
    if wrong_msg: # 오답메세지를 출력했는지 확인  
        if time.time() - wrong_msg_time < wrong_duration:  
            flip_frame = print_korean(flip_frame, "오답!", (width-100, 100), 30, (255,0,0))  
    else:  
        wrong_time = False
```



03-3 구현 과정 - 메인 루프 - PLAYING - 2/5

```
if key == ord('c') or key == ord('C'):
    gray_image = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY) # roi 영역에 있는 값을 Gray로 -> 이진화하려고
    cv2.imwrite('gray_image.jpg', gray_image) # imwrite
    gauss_image = cv2.GaussianBlur(gray_image, (5, 5), 0) # 가우시안 블러
    _, otsu_thread = cv2.threshold(gauss_image, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
    cv2.imshow('otsu_thread', otsu_thread)
```

```
### 1. Morph (모핑)
kernel = np.ones((3,3), np.uint8)
erosion = cv2.erode(otsu_thread, kernel, iterations=1) # 침식방법을 이용해서 확장해서 받아온 이미지를
cv2.imshow('1. erosion', erosion) # erosion이라는 팝업창으로 봅시다.
```

```
### 이미지 자르기
img = erosion
#img = dilation
h, w = img.shape[:2]
crop_size = 280 # 길이 280으로 자르겠다.
cx, cy = w // 2, h // 2 # center 잡기
half = crop_size // 2 # 절반 사이즈
x1, x2 = cx - half, cx + half # x축 좌우로 140
y1, y2 = cy - half, cy + half # y축 위아래로 140
```

```
### 2. 경계면 설정
x1 = max(0, x1)
y1 = max(0, y1)
x2 = min(w, x2)
y2 = min(h, y2)
cropped_img = img[y1:y2, x1:x2]
cv2.imshow('2. cropped_img', cropped_img)
```

01

사용자 입력

if문으로 key == ord('c')로 c를 눌러야 상호작용할 수 있도록 합니다.

02

이미지 가공

이미지 가공을 위한 로직을 실행합니다.
가우시안 블러, OTSU 이진화, Erosion, 이미지 crop 등

03

imshow

이미지 가공 각 단계별로 이미지가 어떻게 되고 있는지 확인할 수 있도록 중간중간 imshow()로 팝업창을 띄웁니다.

03-3 구현 과정 - 메인 루프 - PLAYING - 3/5

```
pad = 50 # 여백 크기  
padded_img = cv2.copyMakeBorder(cropped_img, pad, pad, pad, pad,  
                                 cv2.BORDER_CONSTANT, value=[0, 0, 0])  
  
resized_img = cv2.resize(padded_img, (28, 28))  
  
if len(resized_img.shape) == 3:  
    resized_img = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)  
  
cv2.imshow('4. resized_img', resized_img)  
cv2.imwrite('resized_img.jpg', resized_img)  
  
# 정규화 후 28,28 -> 1, 28, 28, 1로 차원확장  
test_data = resized_img.astype('float32') / 255.0  
test_data = test_data.reshape(1, 28, 28)
```

01

이미지 패딩

copyMakeBorder 함수를 사용해 숫자 주변에 여백(pad)를 추가함으로써, 이미지 리사이징 과정에서 숫자의 외곽 정보가 손실되거나 왜곡되는 것을 방지합니다.

02

이미지 리사이징

학습 모델 규격인 28x28 사이즈에 맞게 리사이징하고, 흑백으로 변환하여 연산에 불필요한 색상 정보를 제거합니다.

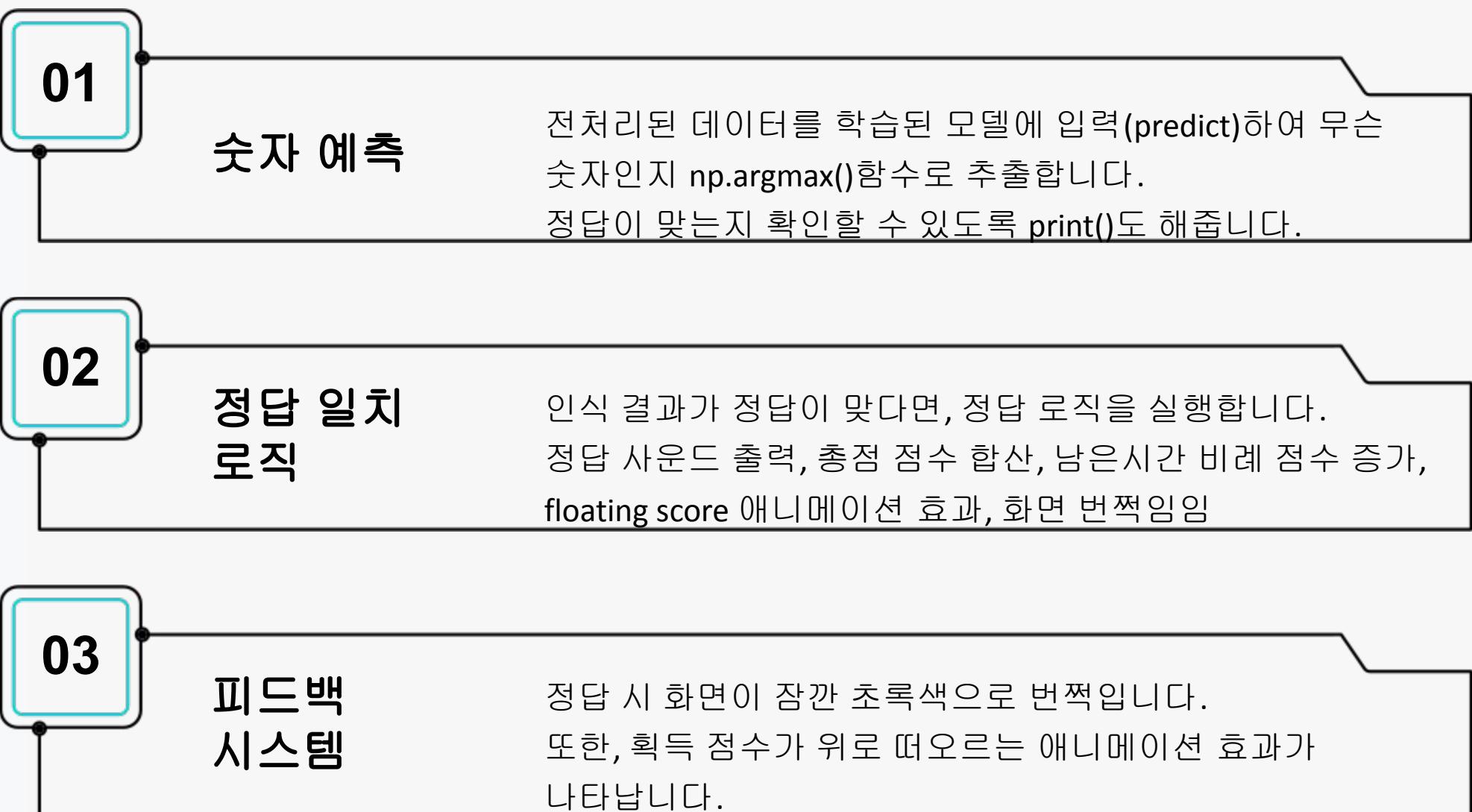
03

데이터 정규화

리사이징한 이미지를 정규화하여 학습 효율을 올리는 효과를 기대합니다.
이미지를 인식할 수 있도록 차원 확장하여 차원을 맞춥니다.

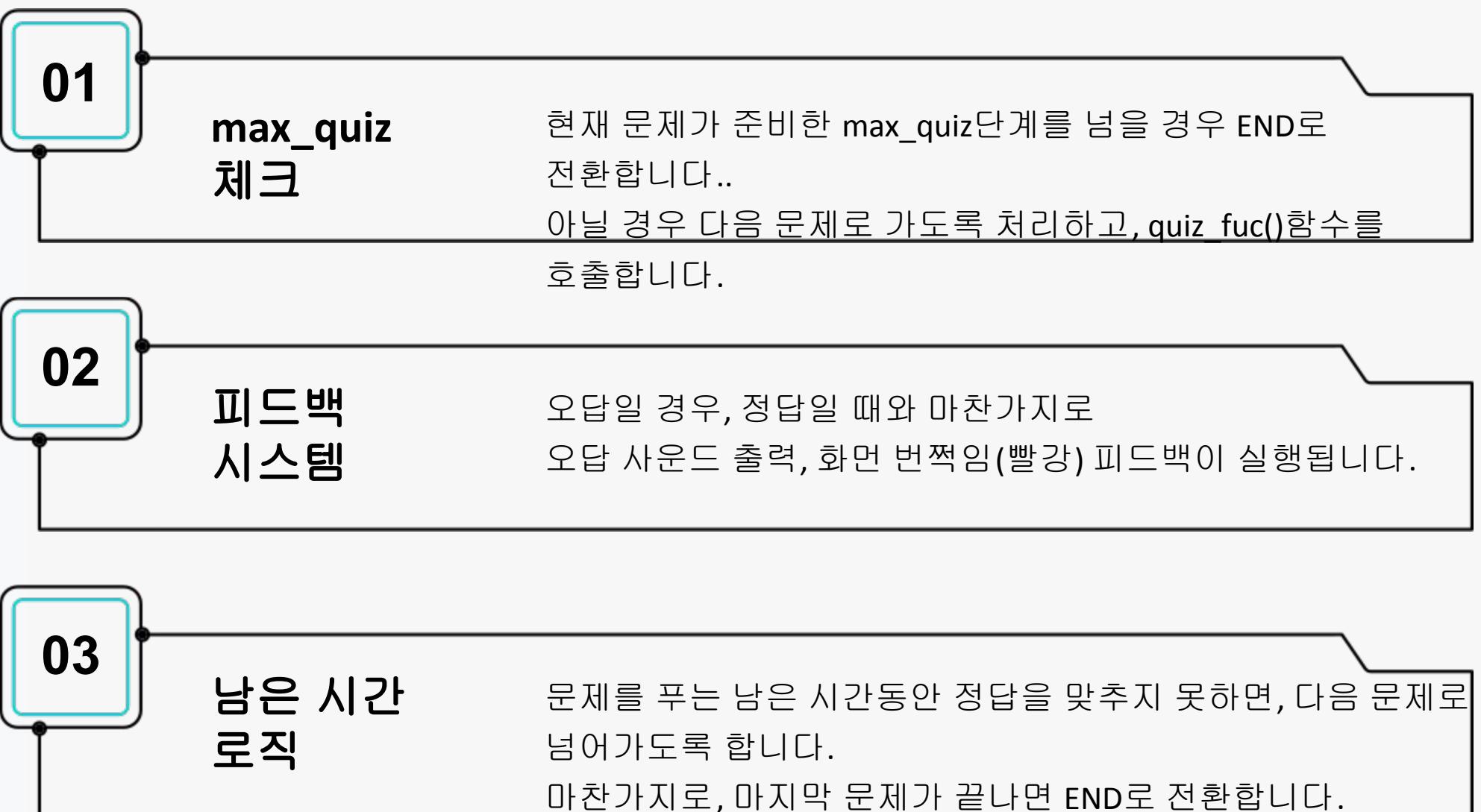
03-3 구현 과정 - 메인 루프 - PLAYING - 4/5

```
try:  
    pred_img = model.predict(test_data , verbose=0)  
    result = np.argmax(pred_img)  
    print(f"인식 결과 : {result}, 정답 : {answer}")  
  
    if result == answer:  
        print("정답!")  
        wrong_msg = False  
        sound_correct.play()  
  
        total_score += remain_time  
        plus_score = remain_time  
  
    # floating score  
    score_effects.append({  
        'val': plus_score,  
        'start_time': time.time(),  
        'x': width - 150, # SCORE 표시 근처  
        'y': 60  
    })  
  
    # 화면 반짝임  
    flash_color = (0, 255, 0) # 초록색  
    flash_time = time.time()
```



03-3 구현 과정 - 메인 루프 - PLAYING - 5/5

```
if quiz_count >= max_quiz:  
    game_mode = "END"  
else:  
    quiz_count += 1  
    quiz_fuc(game_level) # 다음 문제 생성  
    start_time = time.time() # 시간 초기화  
  
else:  
    print("오답!")  
    wrong_msg = True  
    sound_wrong.play()  
  
    flash_color = (0, 0, 255) # 빨간색  
    flash_time = time.time()  
  
    wrong_msg_time = time.time() # 연타해도 마지막 기준 1초 동안 메세지 출력하도록  
  
except Exception as e:  
    print(f"error : {e}")  
  
if remain_time <= 0 :  
    if quiz_count >= max_quiz:  
        game_mode = "END"  
    else:  
        quiz_count += 1  
        quiz_fuc(game_level)  
        start_time = time.time()
```



03-3 구현 과정 - 메인 루프 - END

```
elif game_mode == "END":  
    level_str = "초급" if game_level == 1 else "중급"  
    flip_frame = print_korean(flip_frame, "GAME OVER",  
                             (center_x - 120, center_y - 120), 50, (0, 0, 255))  
  
    flip_frame = print_korean(flip_frame, f"난이도 : {level_str}",  
                             (center_x - 100, center_y - 40), 30, (255, 255, 255))  
  
    flip_frame = print_korean(flip_frame, f"최종 점수 : {total_score:.1f}점",  
                             (center_x - 120, center_y + 10), 35, (0, 255, 255))  
  
    flip_frame = print_korean(flip_frame, "Enter를 눌러 메인으로",  
                             (center_x - 140, center_y + 80), 25, (200, 200, 200))  
  
if key == 13: # 엔터  
    game_mode = "WAITING"
```



03-3 구현 과정 - 메인 루프 - 종료

```
# Floating Score 애니메이션
curr_t = time.time()
for effect in score_effects[:]:
    elapsed = curr_t - effect['start_time']
    if elapsed > 1.0: # 1초 지나면 사라짐
        score_effects.remove(effect)
    continue
```

```
# y값은 위로(-), 투명도는 낮게 계산
move_up = int(elapsed * 60)
# print_korean 함수를 써서 출력
flip_frame = print_korean(flip_frame,
    f"+{effect['val']:.1f}", (effect['x'], effect['y'] - move_up), 30, (0, 255, 255))
```

```
# 화면 번쩍임을 화면에 그리기
if time.time() - flash_time < 0.2: # 0.2초 동안만 번쩍
    overlay = flip_frame.copy()
    cv2.rectangle(overlay, (0, 0), (width, height), flash_color, -1)
    # addWeighted로 투명도 조절 (0.3은 30% 높도)
    flip_frame = cv2.addWeighted(overlay, 0.3, flip_frame, 0.7, 0)
```

```
cv2.imshow(window_name, flip_frame)
if key == 27:
```

```
    break
```

```
cap.release()
cv2.destroyAllWindows()
```

01

Floating Score

코드 상단에서 append한 score_effects를 이 위치(imshow로 그리기 직전) 구현합니다.

정답 시 획득한 점수가 1초동안 위로 떠오르는 애니메이션

효과

02

화면 번쩍임

마찬가지로, 코드 상단에서 전처리만 해놓고, 그리기 직전에 화면 번쩍임 효과를 출력합니다.

03

리소스 해제

ESC 입력 시 루프를 탈출합니다.(게임 종료)

cap.release()와 cv2.destroyAllWindows()로 리소스 반환하고 종료합니다.

04

Thank You

GitHub & Youtube Link

https://github.com/Hoon0185/Tensorflow_OpenCV_Project20260114_20260116

https://youtu.be/AsDR_6ZPi-E

ps. 숫자 9 인식 문제 : 6을 뒤집어서 인식시켜서 -> 9 손글씨로 인식 시 정상동작