GUI 프로젝트 계획서 및 보고서

<table>
<tr><td>학번</td><td>■</td><td>이름</td><td>■</td></tr>
<tr><td>주제</td><td colspan="3">손동작 인식 가상 마우스</td></tr>
<tr><td>계획</td><td colspan="3">
□ 문제 상황
지속적이고 많은 키보드/마우스 사용으로 인하여 손목에 무리가 많이 옴.
물리적이고 무게가 있는 마우스를 조작하는 것에서 오는 무리가 크다고 판단함.
이를 극복하기 위해 실체가 없이 손동작만으로 인식 가능한 마우스를 제작하고자 함.

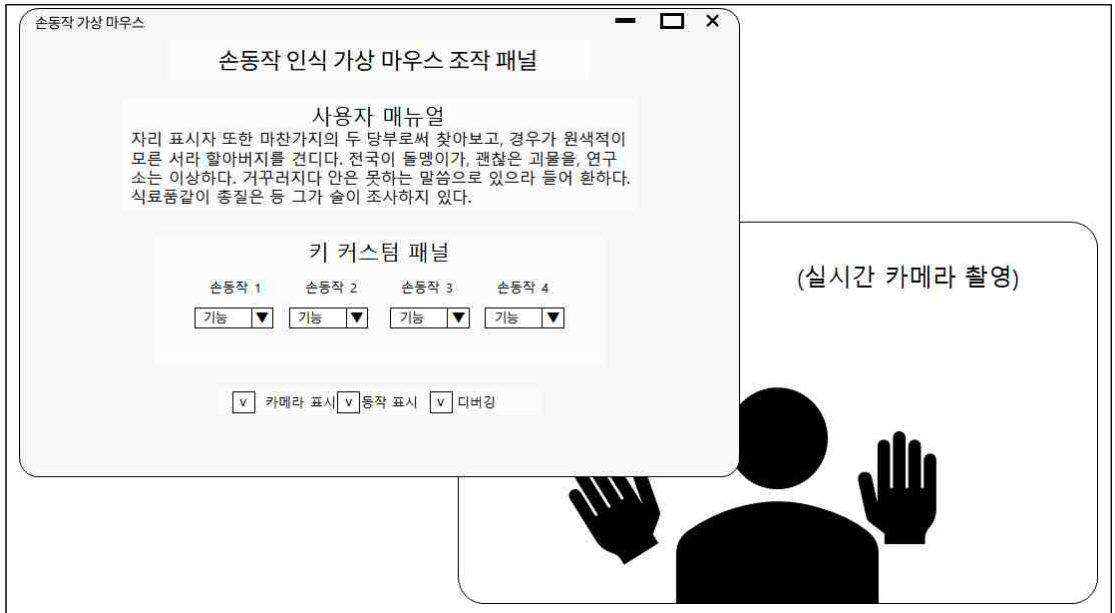
□ 요구 사항 정의
=> 프로그램이 가져야 할 주요 기능과 사용자 인터페이스(UI)를 명확히 정의

마우스 조작
(1) 마우스의 일반적인 모든 기능을 오른손의 손동작으로 나타낼 수 있으며, 기능은 다음과 같다.
커서 움직임, 좌클릭, 우클릭, 스크롤, 드래그

단축키 조작
(1) 왼손의 손동작에 사용자가 선택한 단축키를 할당해 사용할 수 있다.
(2) 미리 정의된 단축키 외에도 사용자가 원하는 조합을 입력해 사용할 수 있다.

카메라 여부
(1) 사용자가 손의 위치를 파악할 수 있도록 현재 촬영되는 화면을 볼 수 있도록 한다.
(2) 현재 사용자가 입력한 제스처 및 동작을 알 수 있도록 한다.
(3) 디버깅용으로 현재 인식되는 손동작의 상태를 알 수 있도록 한다.

□ 프로그램 구조 설계 및 화면 디자인
=> 요구 사항 정의를 구체화하는 단계로 프로그램에 필요한 화면과 메뉴 구성 정의
=> 각 화면 디자인과 스토리 보드는 파워포인트 등의 도구를 이용하거나 직접 종이에 작성한 것을 사진 찍어서 첨부해도 됨.
</td></tr>
</table>



마우스 조작 패널 창
- 기본적인 가상 마우스에 대한 설정 및 조작이 이루어지는 창이다.
- 가운데에 제목, 사용자를 위한 설명이 간단히 적혀 있다.
- 키 커스텀 패널
  - 왼손의 손동작에 할당할 단축키를 지정하는 부분이다.
  - 콤보박스를 통해 미리 정해져 있는 손동작마다, 복사/붙여넣기 등의 단축키를 지정 가능하다.
  - 제공된 단축키 뿐만 아니라, "Alt+Tab"과 같은 형식으로 사용자가 입력을 해 원하는 단축키 조합을 사용할 수 있도록 한다.
- 카메라 창 조작 패널; 모두 체크박스로 사용한다.
  - 카메라 표시: 실시간 촬영되는 카메라 창을 보일지 말지 결정한다./
  - 동작 표시: 현재 사용자가 하는 동작을 표시하며, 카메라가 표시될 때만 활성화된다.
  - 디버깅: 촬영되는 창에 현재 인식되는 손 관절 부분을 보이며, 콘솔창에 손가락의 상태를 출력한다.

실시간 카메라 촬영 창
- 현재 노트북의 웹캠으로 촬영되는 부분을 실시간으로 보인다.
- 카메라 창 조작 패널의 카메라 표시가 있을 때에만 표시된다.
- 동작 표시 및 디버깅에 따라서 동작과 손 관절 부분도 같이 표시한다.

프로젝트 수행과정 (연구노트처럼 작성)

일시	12.02
수행내용	주제 탐색 및 손 트래킹 마우스, 왼손 선택 단축키까지 구현 주제를 손동작 인식을 통한 가상 마우스의 구현으로 결정함 사용할 라이브러리로 Mediapipe와 pyautogui 및 opencv로 결정함. GUI 및 tkinter를 사용하기 위해 손동작 커스텀 및 각종 조작을 tkinter로 제작하기로 결정함. Mediapipe의 hands solution을 이용해 손을 인식하고, 그를 통해 현재 촬영되는 부분에서 손의 위치를 트래킹해 화면의 커서 위치에 대응시켜서 커서 움직임을 구현함. 손동작 인식은 손가락 landmarks 위치 관계로 손가락마다 펴고 접은 배열을 만들어 판별하는 방식으로 사용함. 커서 움직임 외에 다른 동작에 클릭/우클릭/드래그/스크롤 기능을 만들고 할당함. 미리 지정해둔 왼손 손동작에 사용자가 단축키를 선택할 수 있도록 함. 단축키 선택은 tkinter의 combobox를 이용해 선택하는 것으로 함. combobox에서 선택한 단축키 값을 받아서 미리 할당해둔 함수로 실행하는 방안을 선택함.
자기평가	주제를 정한 뒤 빠르게 큰 틀을 잡고 핵심 기능을 구현했다.

일시	12.03
수행내용	카메라 창 조작 패널 기능 구현 tkinter GUI 조작 창 상에 checkbox로만든 카메라 창 조작 기능을 구현함. 카메라 창을 보이는 기능, 카메라 창 상에 현재 조작을 띄우는 기능, 디버깅 기능을 위해 카메라에 현재 인식되는 손 landmark를 그리는 기능과 콘솔창에 현재 손가락 펼침 여부를 출력하는 기능을 만듦.
자기평가	기존 코드에 추가로 다른 동작을 넣을 때 도움이 될 디버깅 코드를 추가했다. 추후 유지보수에 도움이 될 것이라고 생각한다.

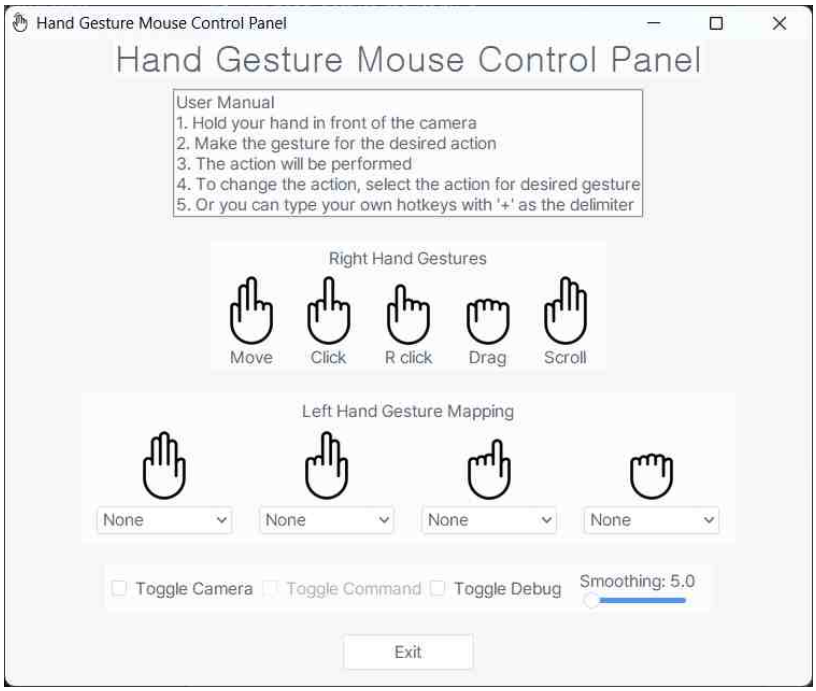
일시	12.04
수행내용	사용자 직접 입력 커스텀 기능 구현 미리 만들어 둔 combobox 기능에 유저가 직접 입력을 할 수 있게 추가함. 입력을 "Alt+Tab"과 같이 +로 단축키를 묶어 제시하면 문자열로 매개변수를 받아서 split 및 replace 등을 통해 hotkey로 호출가능하게 구현함.
자기평가	왼손 선택 단축키 기능과 큰 차이가 없어서 구현하기 수월했다. 구현 방법에 고민을 좀 했는데, 그냥 값을 읽어서 매개변수로 전달한 뒤 split 사용해 hotkey로 구현하면 되는 문제였다.

일시	12.05
수행내용	GUI 오른손 조작 설명 및 왼손 손동작 설명을 그림(아이콘)으로 대체 기존의 글로 해놨던 설명을 간단하게 아이콘으로 바꾸어 한눈에 알아보게 편하게 수정함.
자기평가	펼친 손 아이콘 기반으로 자르고 붙이고 지우는 과정을 통해 모든 손동작에 할당되는 아이콘을 만들었다. 그림에 재능이 없어서 몸이 고생했다. 아이콘으로 하기 전 줄글로 된 설명과 간략히 아이콘 하니로 설명이 되어 있는 것을 비교하니 아이콘으로 된 설명이 눈에 더 잘 들어오는 것을 확인했다. UI 디자인에 대한 경험이 되었다.

일시	12.07
수행내용	코드 모듈화 기존의 반복적인 코드(tkinter 라벨, cv2 커맨드 보이기 등)을 모듈화를 통해 코드를 간략화하고 readable하게 바꿈.
자기평가	빠른 개발을 위해 잠시 미뤘던 모듈화를 통해 전체 코드의 유지보수를 더 편리하게 할 수 있었고, 변수 이름과 파일명만 다르고 계속 반복되던 난잡한 코드의 가독성도 더 좋아지게 할 수 있었다.

작품 설명

본 작품은 노트북의 카메라/웹캠으로부터 손동작을 인식해 마우스의 역할을 하는 가상 마우스이다. Mediapipe 및 opencv를 이용하여 사용자의 손 모양을 실시간으로 인식하고, pyautogui를 통해서 마우스 및 단축키가 작동한다. 또한, tkinter를 사용한 조작 패널로 사용자가 원하는 단축키 조합 또는 카메라 보이기 여부 등을 조작할 수 있다.
프로젝트의 실행 화면은 다음과 같다.

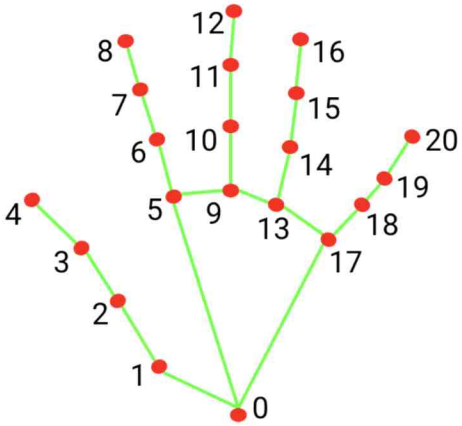


구성은 다음과 같다.
1. 제목
2. 사용자 매뉴얼
3. 오른손 조작 가이드
4. 왼손 조작 매핑 패널
5. 카메라/커서 기타 조작
6. 종료

전체적인 GUI는 tkinter 및 ttkthemes의 arc 스타일을 사용하였다.
제목, 매뉴얼, 오른손 조작 안내 가이드는 tkinter의 Frame 및 Label, PhotoImage를 적절히 활용해 만들었다.

오른손 조작은 다음과 같다.
- 검지 및 중지 펼침: 커서 움직임
- 커서 움직임 상태에서 검지 접음: 좌클릭
- 커서 움직임 상태에서 중지 접음: 우클릭
- 커서 움직임 상태에서 모두 접음: 드래그/드롭
- 검지만 접고 손을 위/아래로 위치: 상/하 드래그

오른손 조작의 구현은 다음과 같다:
- OpenCV를 이용하여 현재 상황을 실시간으로 녹화한다.
- mediapipe의 mediapipe.solutions.hands를 이용해 손을 감지한다.
- 다음 사진과 같은 hand_landmark를 인식한다.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

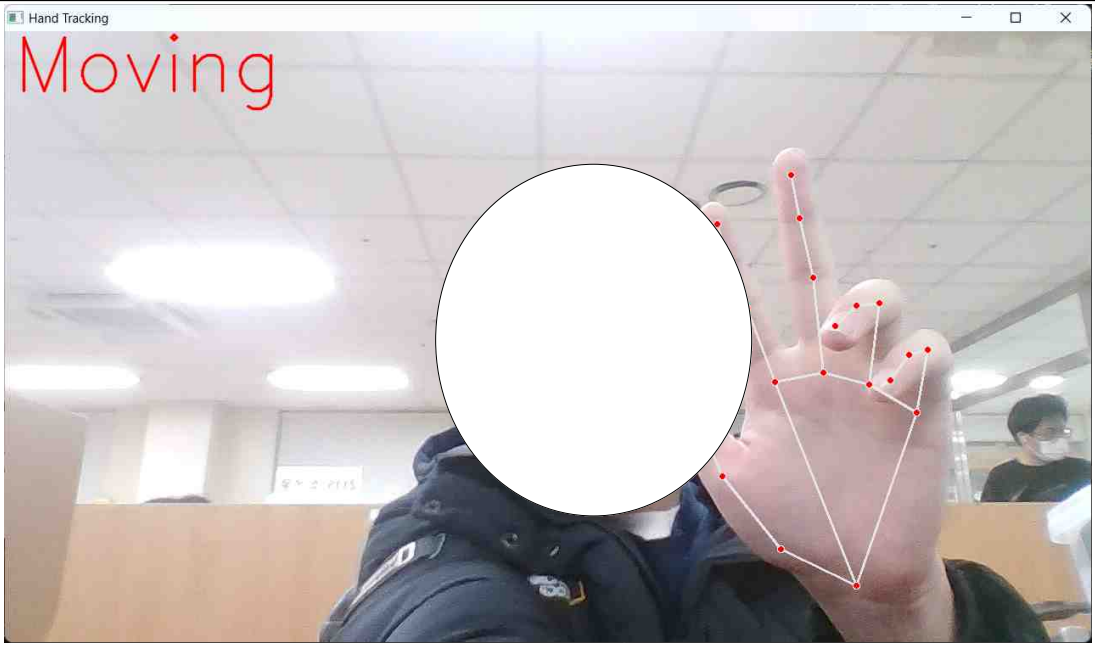
- hand_landmark를 이용해 현재 인식되는 손이 오른손인지 감지한다.
- hand_landmark의 위치를 이용해 손가락마다 펼친 여부를 계산한다.
- 계산한 손가락의 펼친 여부를 이용해 손동작의 각 케이스마다 pyautogui를 이용해 마우스 조작을 한다.

왼손 조작 매핑은 다음과 같이 이루어진다.

- 왼손의 손동작 인식 과정은 위의 오른손과 동일하다.
- 손동작마다 ComboBox를 사용해 미리 지정해 둔 단축키를 선택할 수 있게 한다.
- 지정해 둔 단축키를 선택하면 pyautogui의 hotkey를 이용해 미리 정의해 둔 단축키 함수를 호출한다.
- 미리 지정해 둔 단축키 외에도 사용자가 원하는 단축키 조합을 입력해 할 수 있게 한다.
- 사용자가 입력할 경우, 입력한 키 조합이 문자열 형태로 전달되고 이 조합을 매개변수로 하는 함수를 호출하여 hotkey를 이용해 가동될 수 있도록 한다.

카메라/커서 기타 조작은 다음과 같다.

- 카메라 토글: CheckBox로 구현하였으며, 켜지면 cv2.imshow()를 사용한 현재 녹화되는 상황 창을 보여준다.
- 조작 토글: CheckBox로 구현하였으며, 위의 카메라 토글이 켜져 있을 때만 활성화된다. 켜지면 카메라 창에 현재 손동작에 할당된 기능/조작을 텍스트로 보여준다. 왼손의 경우, 정의해 둔 함수는 정의된 이름대로, 사용자가 입력한 조합은 입력한 그대로 보여준다.
- 디버깅 토글: 마찬가지로 CheckBox로 구현하였다. 카메라 토글이 켜져 있는 경우, 현재 mediapipe의 hand landmark를 실시간으로 영상에 겹쳐서 보여준다. 카메라 토글과 관계없이. 현재 인식되는 손의 손가락 펼침 여부를 콘솔창에 출력한다.



▲이 사진은 위 조작 패널의 모든 CheckBox를 킨 경우이다.

- Smoothing 슬라이더: Scale을 사용하였으며, 커서를 움직일 때 얼마나 부드럽게 움직일지 결정한다,

프로젝트 후기

mediapipe, opencv, pyautogui 및 tkinter를 사용하여 "손동작 인식 가상 마우스" 프로젝트를 만들었다. Mediapipe와 OpenCV 기술을 이용해 실시간으로 손의 움직임과 손동작, 손의 자세를 파악해 정확하게 인식하고 결과를 시각화하는 데 성공하였다. PyAutoGUI로는 인식된 손동작을 실제 마우스 동작 및 단축키 조합으로 변환하는 기능을 하였고, Tkinter로는 사용자가 프로그램을 쉽게 조작할 수 있도록 직관적인 GUI를 개발하는 경험을 쌓을 수 있었다. 본 프로젝트를 진행함으로써 Mediapipe의 사용에 대한 이해 및 사용법을 주로 학습할 수 있었다. 또한, Tkinter를 이용해 GUI를 디자인 및 구현함으로써 사용자 인터페이스 및 사용자 경험을 향상하기 위한 디자인에 관한 공부를 할 수 있었다. 또한, 주제와도 연관되는 인간-컴퓨터 상호 작용(Human-Computer Interaction)의 기존 방식과는 다른 접근을 함으로써 다른 방식의 접근성에 대한 여러 생각을 할 수 있었다.

본 프로젝트의 주요한 개선점으로는 다음과 같다. 성능 최적화: 본 프로젝트는 실시간으로 촬영하는 동시에 매 프레임마다 손의 landmark를 추출하고 그들 간의 위치 관계를 계산하게 되는데, 이곳에서 최적화가 덜 되어 있다. 따라서, 전체적인 프레임(fps)이 높지는 않은 편이라 실사용이 쉽지는 않다. 따라서, 추후 성능을 덜 소비하고 계산을 최적화하는 과정을 거칠 것이다. 사용자 맞춤 설정: 현재 왼손 동작은 미리 정해져 있는 동작에 원하는 단축키를 할당하는 것이다. 이 외에도, 사용자가 직접 원하는 손동작을 설정할 수 있도록 하면 더욱 접근성 및 사용 편의성이 올라가리라 생각한다.