3. 구현 완성도 개선

과목명 : 휴먼컴퓨터인터페이스

학번 : 2012726030

학과 : 컴퓨터소프트웨어학과

이름 : 유가람

제출일 : 2018. 05. 25

■ 과제 요구사항 체크리스트

항목	구현 유무
사용자 정의 함수에 대한 그래프 그리기	O
둘 이상의 함수에 대한 그래프 중첩 지원	O
원점, 좌표축, 눈금 표시	O
그래프 표시 영역 조절 가능	O

■ 오픈소스 라이브러리 의존성

API	디렉토리	파일
jQuery	dist/js	jquery.min.js
bootstrap	dist/css	bootstrap.min.css
	dist/js	bootstrap.min.js
mathjs	dist/css	style.css
	dist/js	math.js
		calc.js
remodal	dist/css	remodal.css
		remodal-default-theme.css
	dist/js	remodal.js
font-awesome	dist/css	font-awesome.css
	dist/fonts	fontawesome.otf
		fontawesome-webfont.eot
		fontawesome-webfont.svg
		fontawesome-webfont.ttf
		fontawesome-webfont.woff
		fontawesome-webfont.woff2
plotly	dist/js	plotly.min.js
		plotly.draw.js
numpad	dist/css	numpad.css
	dist/h	numpad.js
		numpad.settings.js

■ 사용성 향상에 기여하는 핵심적인 상호작용 방식

□ 그래프를 직관적이게 볼 수 있다.

- ‘plotly’ 라이브러리를 사용하여 그래프를 나타내는 데에 효과적이다.

□ 정의역 구간을 손쉽게 제어할 수 있다.

- ‘numpad’ 라이브러리를 사용하여 숫자를 입력할 수 있는 창을 띄워 상호작용에 도움을 준다.
- 이 부분은 슬라이더 컨트롤을 이용하여 해결하려 했으나, 사용자가 원하는 값을 알 수 없으므로 텍스트 입력 방식에서 마우스만 이용하여 입력할 수 있는 방법 중 ‘numpad’를 택하게 되었다.

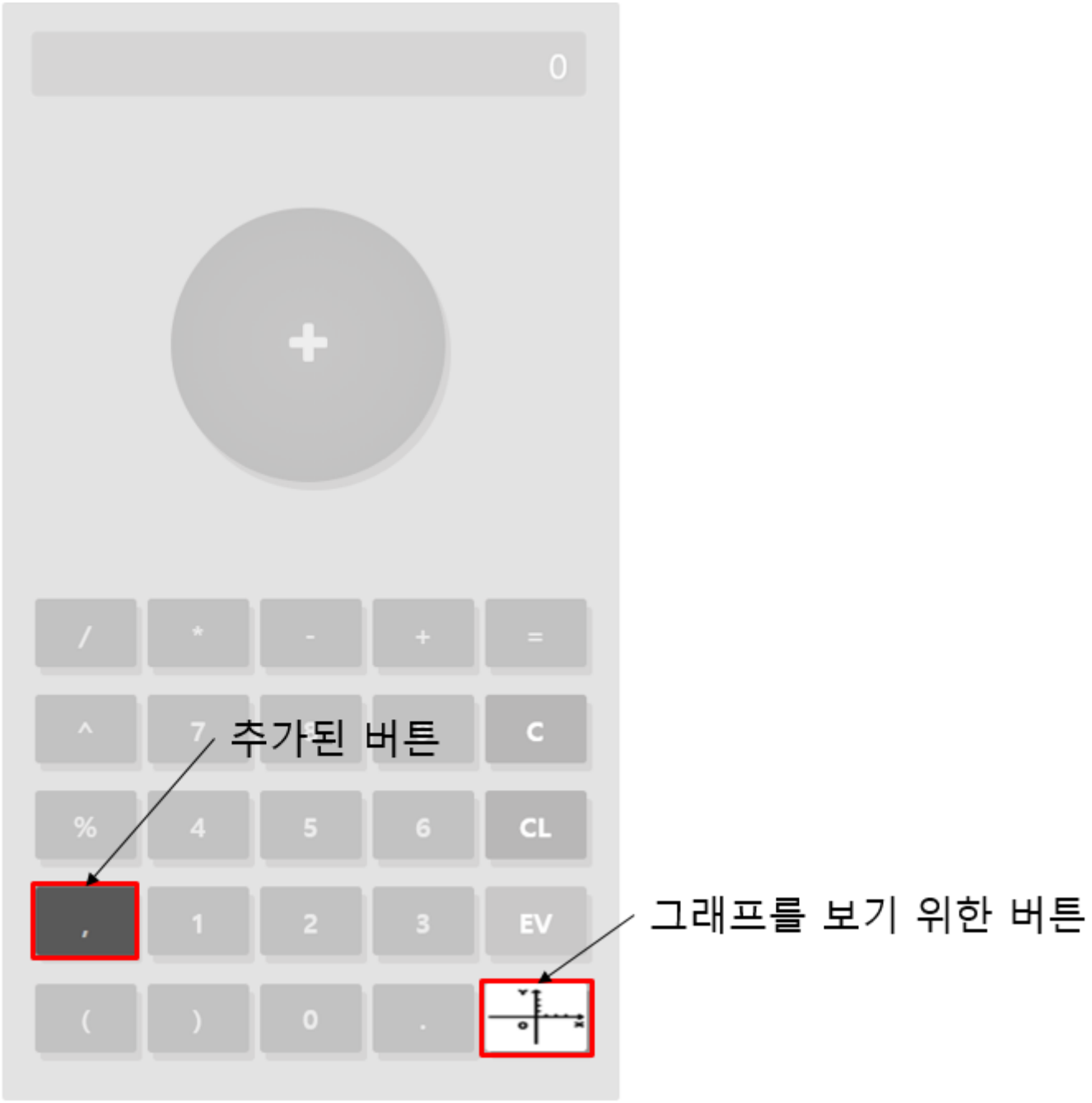
■ 기존 계획으로부터 변경된 부분과 그 이유

□ 프로그래머용 기능

■ 과제2에서도 완성하지 못한 진법 변환을 이번에도 시간적 이유로 배제하게 되었다.

■ 새로운 사용자 인터페이스의 구성 요소 및 사용 방법

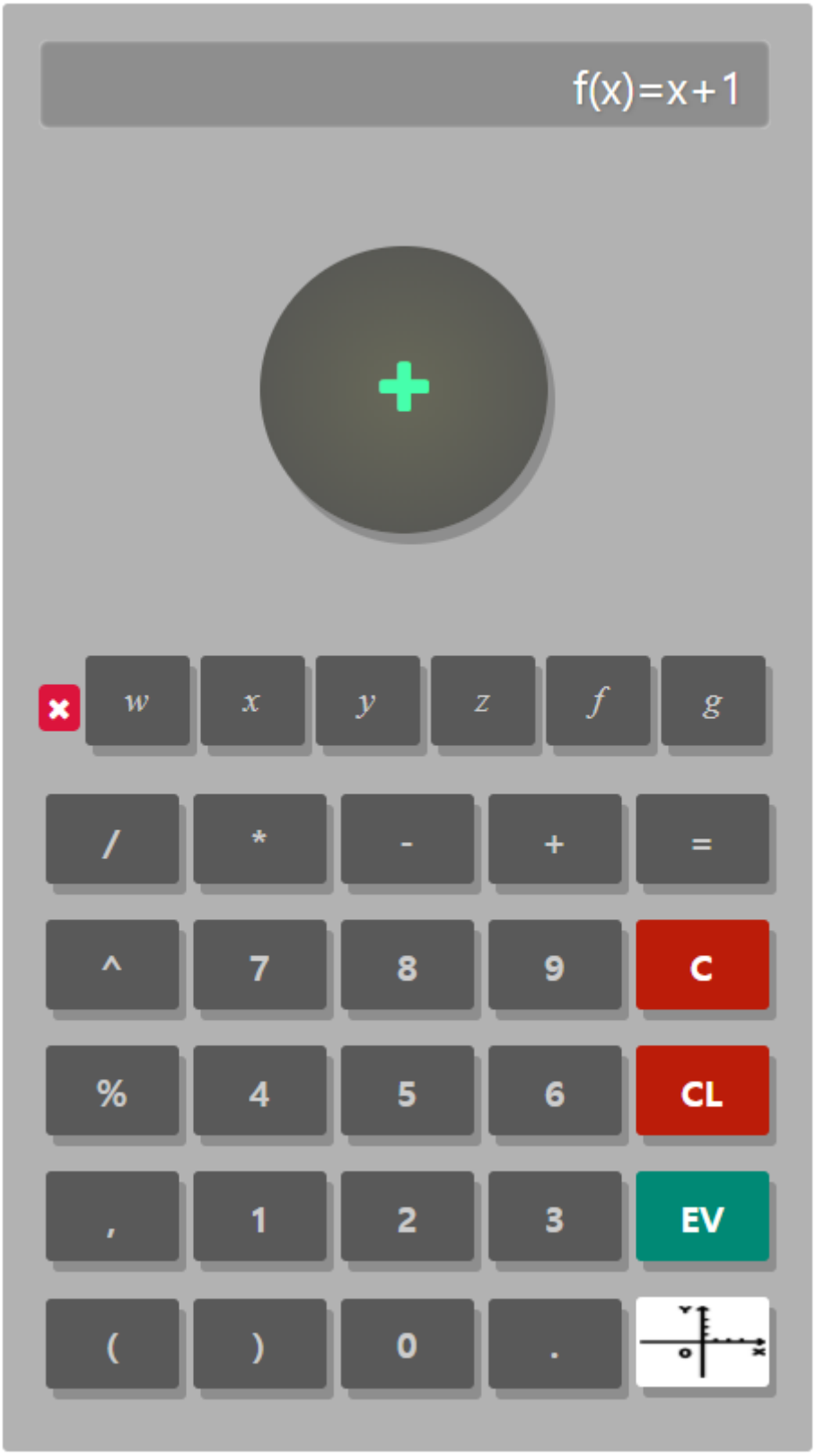
□ 계산기 구성 요소



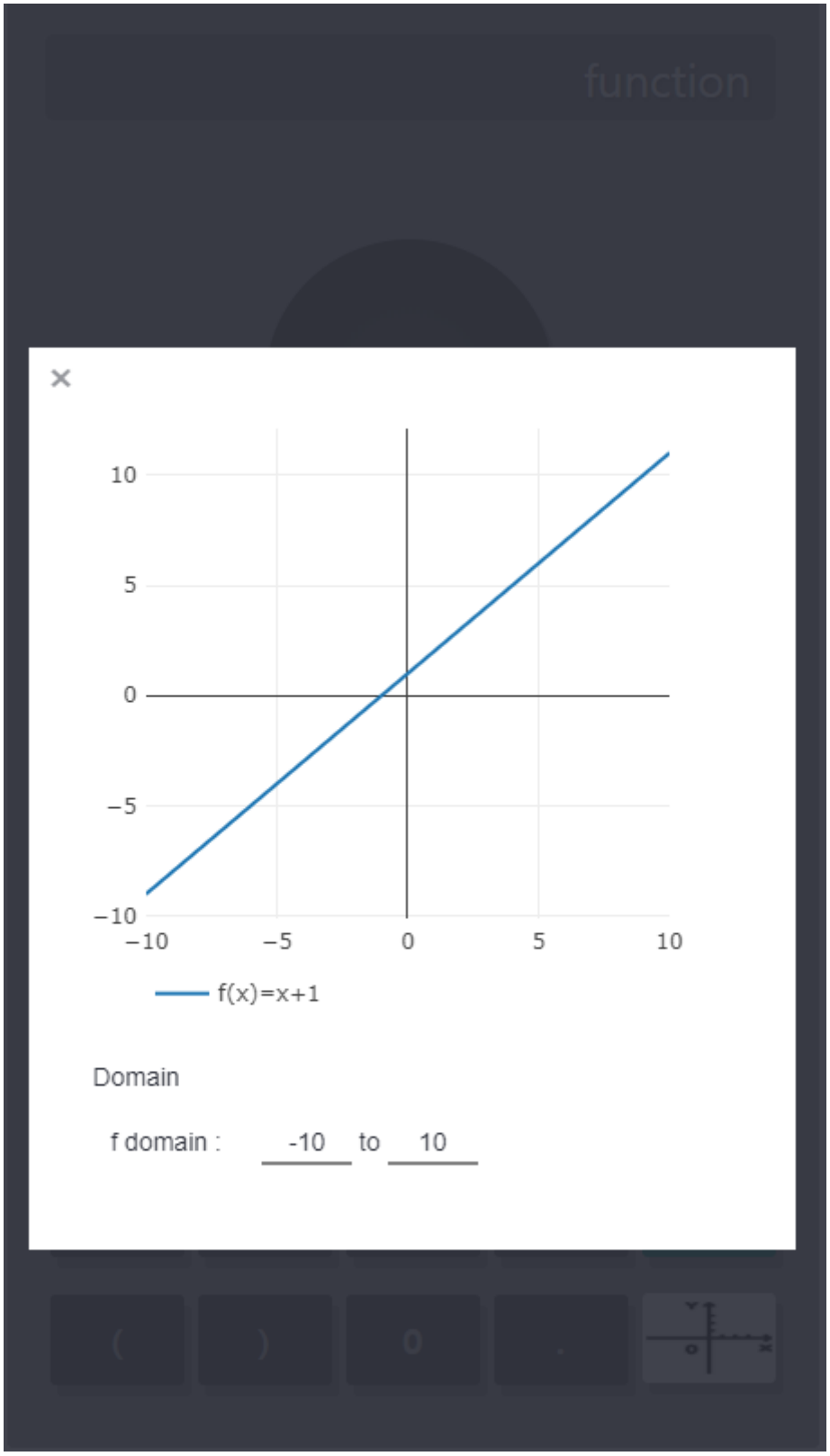
□ 기존 UI 설계 실패로 추가된 ',' 콤마 버튼

□ 그래프를 보기 위한 버튼 사용 방법

1) 함수 정의

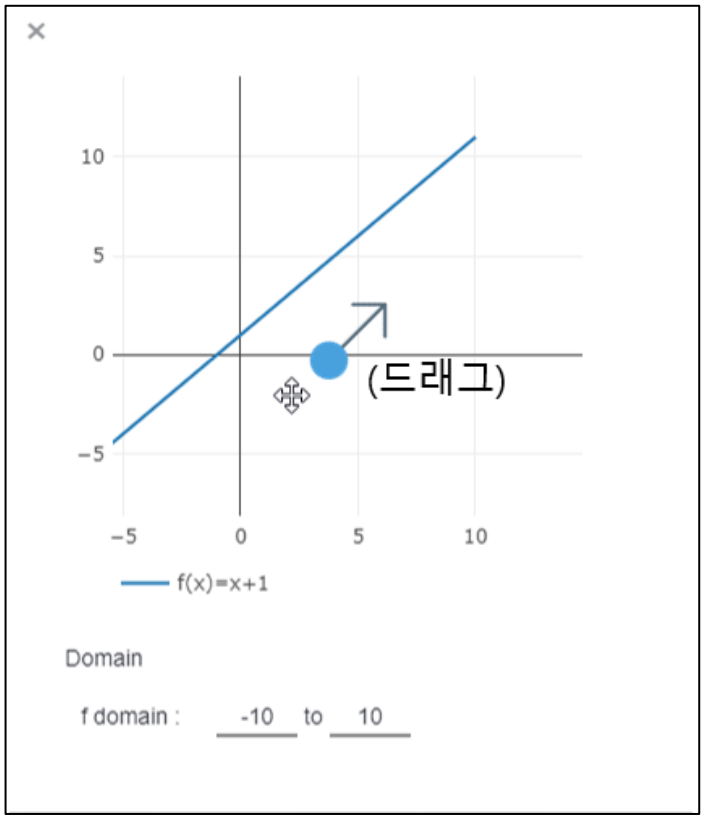


2) 그래프 버튼을 누른다..



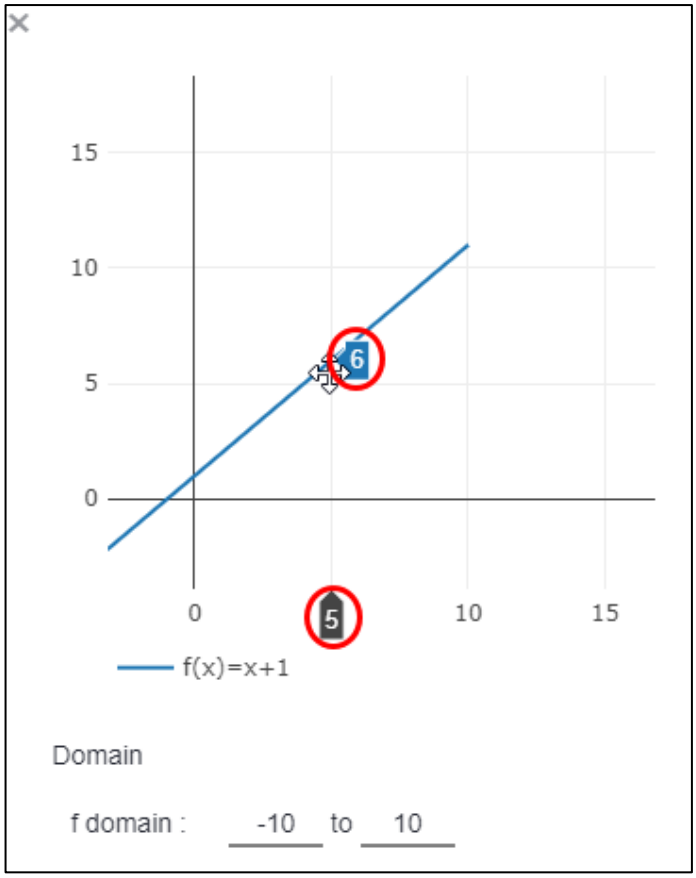
화면 중앙에 그래프가 표현되며, 기본 정의역의 범위는 -10 부터 +10까지 이다.

3) 원점 이동



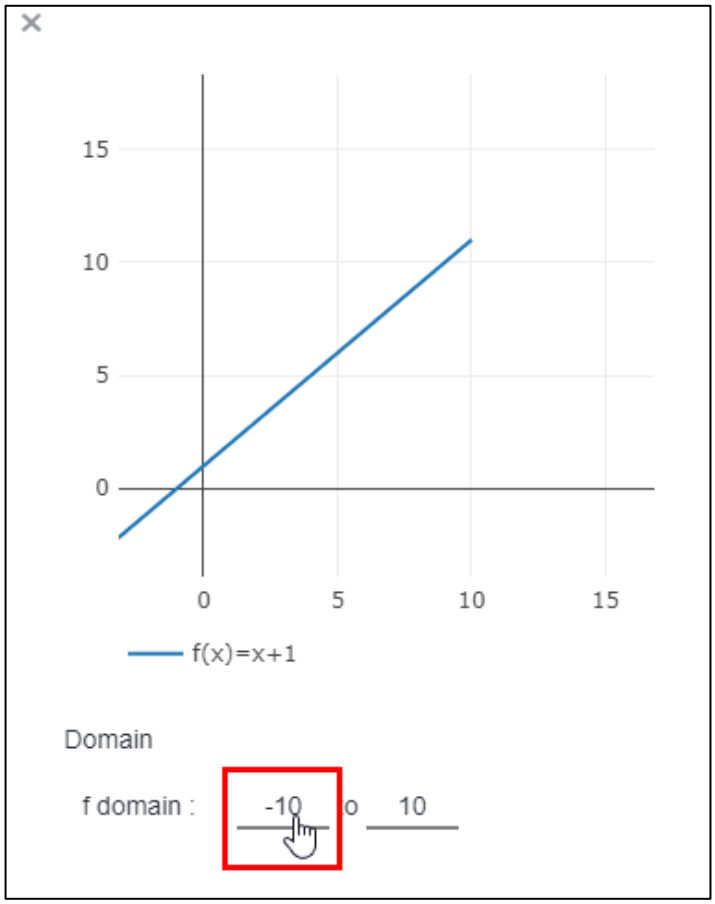
그래프를 드래그하게 되면 손쉽게 원점을 이동할 수 있다.

4) 좌표 정보 보기

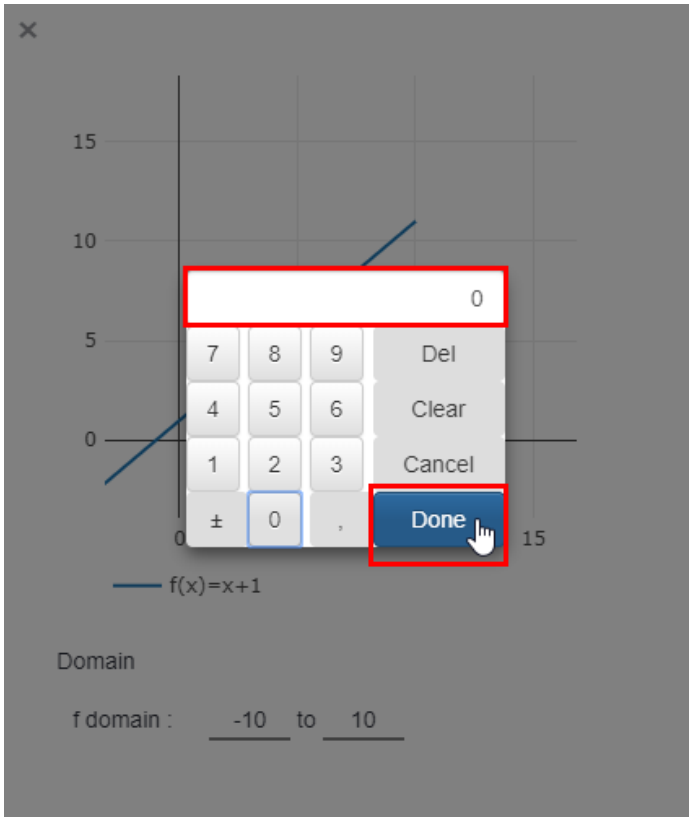


커서를 그래프위에 올리게 되면 하단에는 x값과 그래프 위에는 y값이 표현된다.

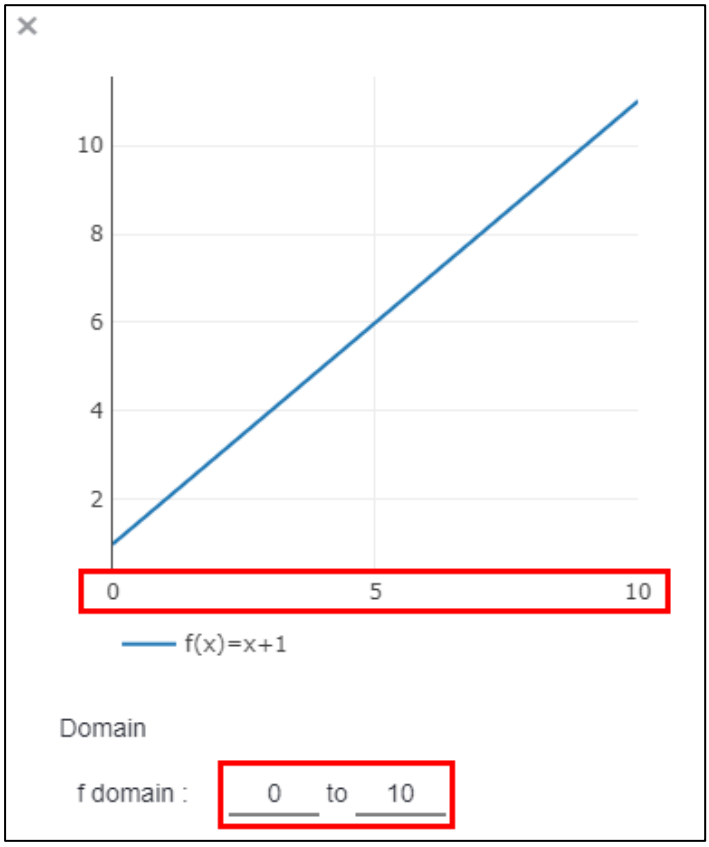
5) 정의역 구간 조정하기



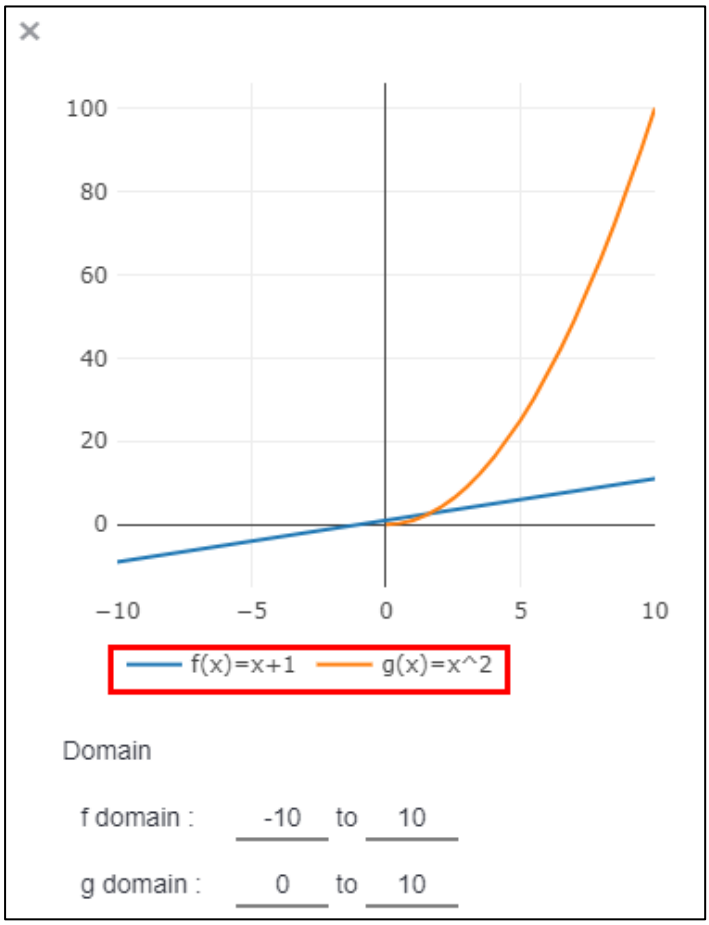
변경하고자 하는 정의역의 시작 범위 혹은 종료 범위에 커서를 올려 클릭한다.



클릭하면 위와 같이 숫자를 입력할 수 있는 'Numpad' 창이 나타나며, 숫자를 입력 후 'Done'을 누른다.

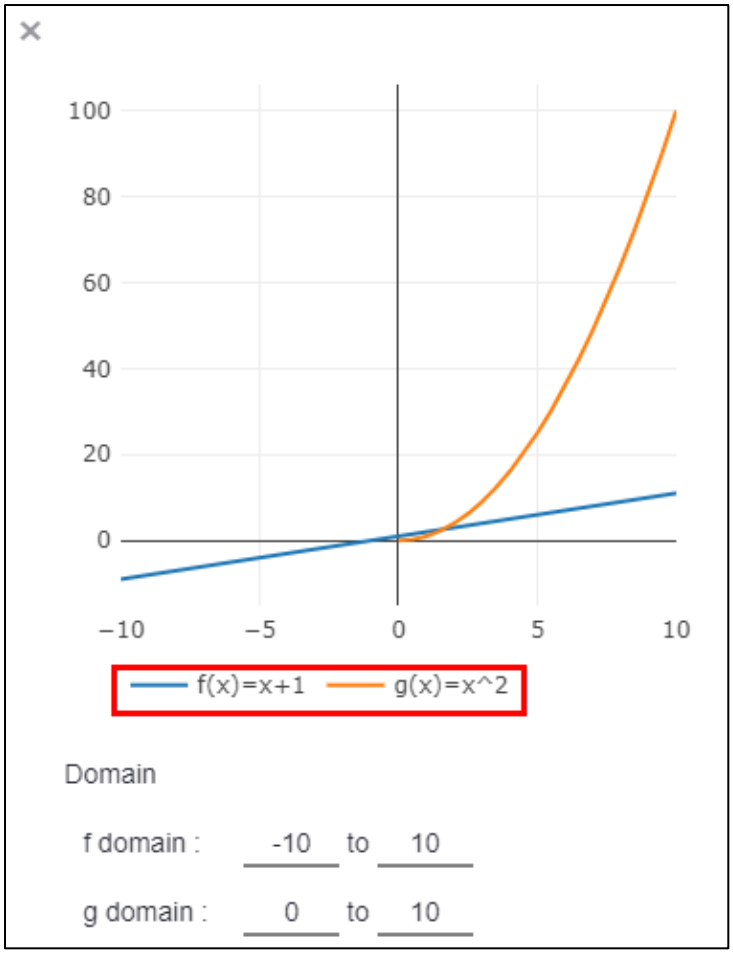


변경된 정의역 구간을 확인할 수 있다.

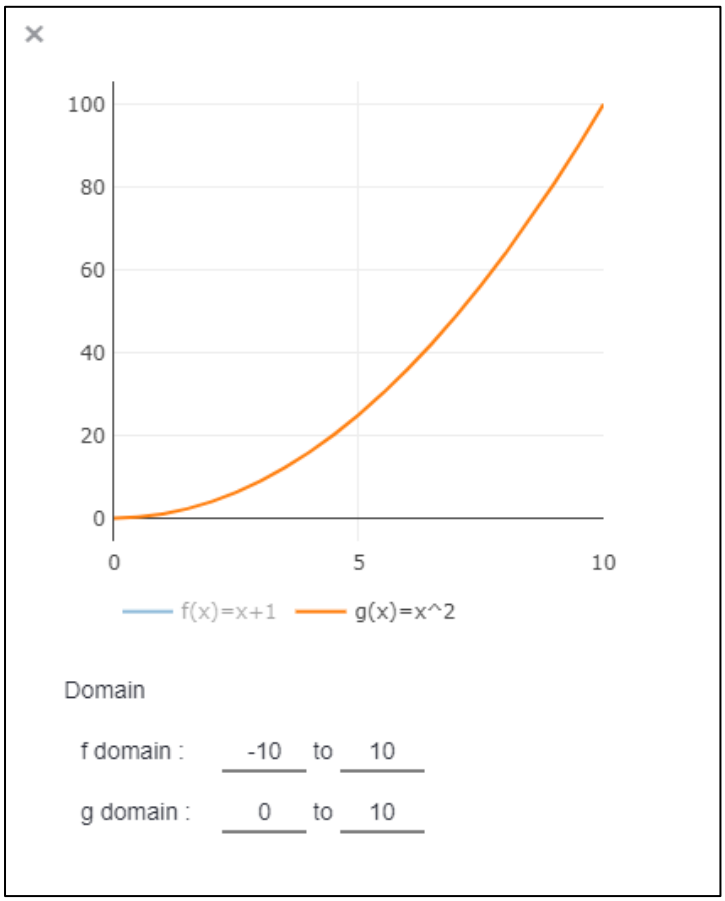


그래프는 최대 두개까지 나타낼 수 있으며,

인자는 'x'로만 이루어진 함수만 나타낼 수 있다.



위 빨간 네모는 그래프의 범례를 나타내는데,
둘 중 하나를 클릭하면 좌표평면 상에서 숨길 수 있다.

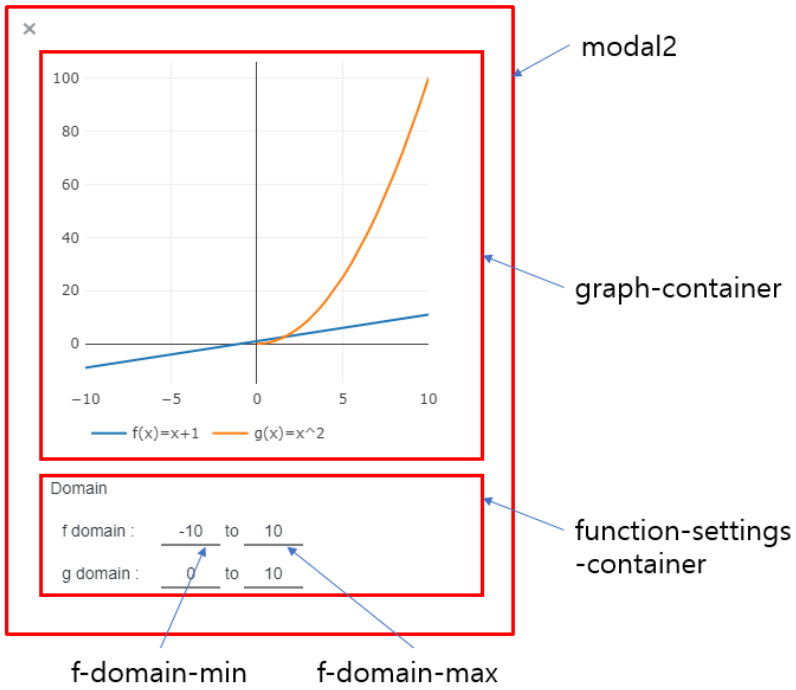


위 예제는 'f(x)'를 숨긴 결과이다.

■ 특징적인 상호작용 방식들에 대한 구현 방법

□ 그래프 표현하기

■ ‘EV’버튼을 통해 함수가 정의가 되면, 계산기의 하단에 ‘display: none’ 속성을 가진 컨트롤에 함수를 저장한다.



■ 과제2에서 사용한 ‘Remodal’ 라이브러리를 활용하여 새로운 창을 띄운다.

■ 그래프 모달 폼(‘modal2’)의 구성은 위와 같으며, ‘f’와 ‘g’의 정의 유무에 따라, 그래프의 표현 개수가 달라진다.

■ ‘f-domain’과 ‘g-domain’이 나타나며, 그래프 또한 정의 유무에 따라 나타난다.
정의 유무는 계산기 하단에 ‘display: none’ 속성을 가진 컨트롤에 저장된 값으로 판단한다.

```
// f function
expr = math.compile(f_define);
var f_xValues = math.range( f_domain[0], f_domain[1] + gap, gap).toArray();
var f_yValues = f_xValues.map(function (x) {
    return expr.eval({x: x});
});

// trace f
// render the plot using plotly
var trace = {
    x: f_xValues,
    y: f_yValues,
    type: 'scatter',
    name: f_full_define
};

data.push( trace );
```

- 'math.js' 라이브러리를 이용하여 함수 'f'와, 'g'의 정의역과 치역을 구한다.

```
var layout = {
    //hovermode:'closest', // hover시, 좌표 모드( 단일 그래프에서만 적용 )
    dragmode: 'pan',       // 드래깅 모드
    autosize: false,
    width: 350,
    height: 350,
    margin: {
        l: 30,
        r: 30,
        b: 20,
        t: 10,
        pad: 4
    },
    paper_bgcolor: '#ffffff',
    plot_bgcolor: '#ffffff',
    showlegend: true,              // 범례 표시
    legend: {"orientation": "h"}    // 범례 위치
};

Plotly.newPlot('graph-container', data, layout, {displayModeBar: false});
```

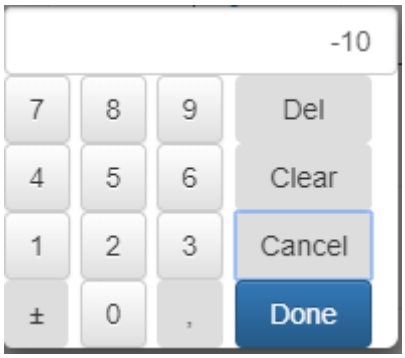
- 그래프의 레이아웃을 설정하고, 그래프의 표현 범위와 레이아웃을 인자로 넘겨 'Plotly.newPlot'으로 그리게 된다.

```
// 정의역 구간 변경 시 이벤트 발생
$('.domain-input').each(function( index, element ) {
}).on('change', function() {
    draw();
});

// 그래프 버튼 클릭 이벤트 발생
$('.key.show-graph').click(function(e) {
    draw();
});
```

- 그래프를 그리는 시점은 정의역 구간의 변동 혹은 그래프 버튼을 클릭할 때 그리게 된다.
- 함수의 (재)정의 시에도 다시 그려야 하지만, 그래프 버튼을 누른 시점이 (재)정의 후이기 때문에 (재)정의에 대한 이벤트는 따로 등록하지 않았다.

□ 정의역 구간 값 입력하기



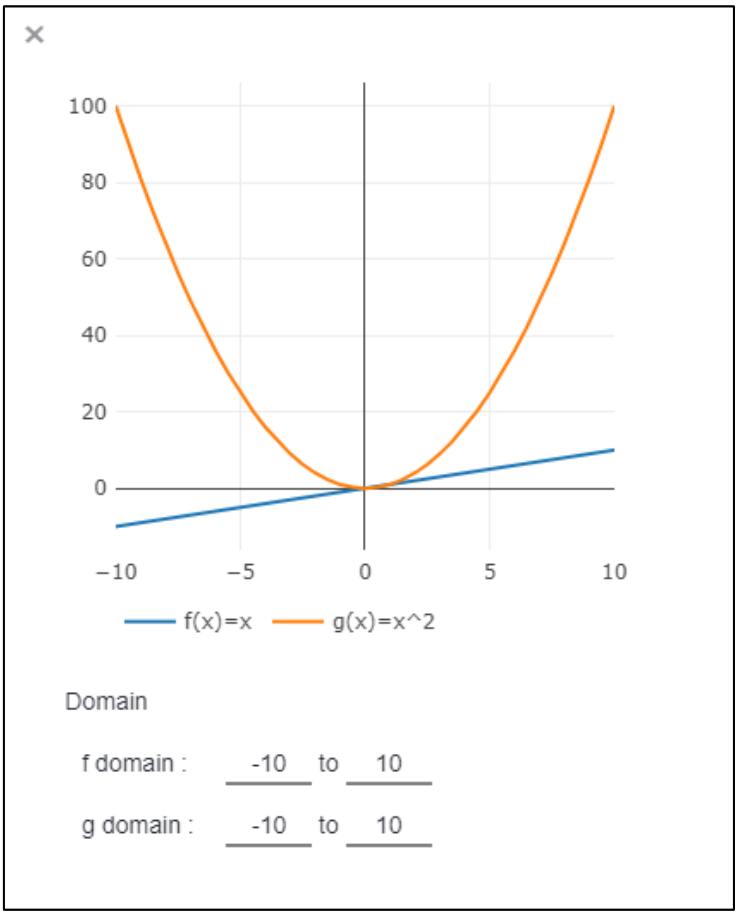
- 'numpad' 라이브러리를 사용하여 숫자를 입력할 수 있는 모달리스 창을 띄우게 된다.

```
$(document).ready(function(){
    $('.domain-input').each(function( index, element ) {
        $(this).numpad();
    });
});
```

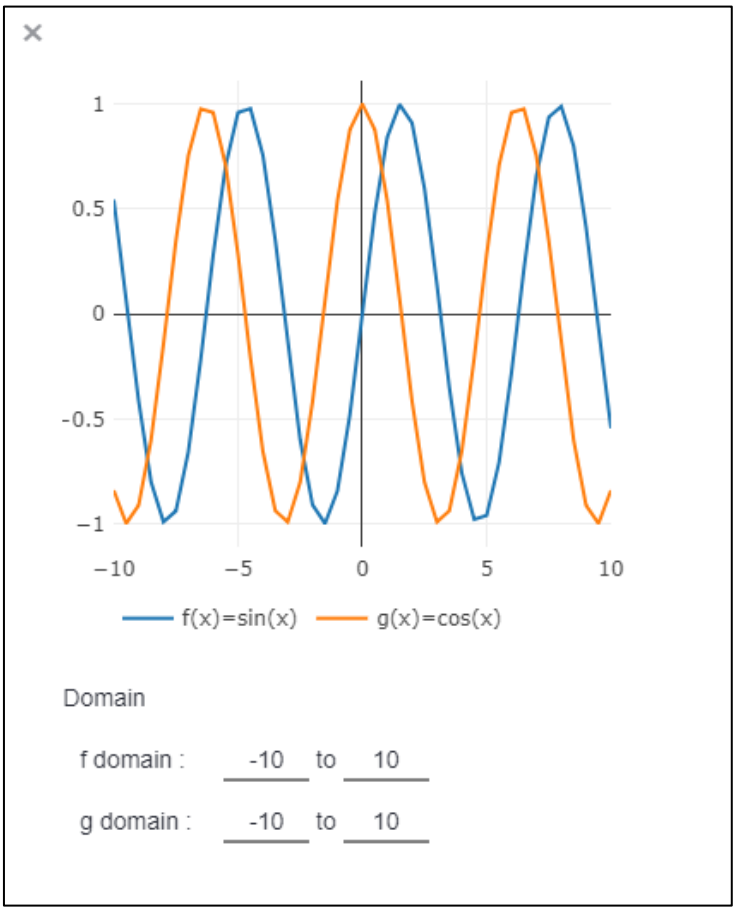
- 복잡한 설정 없이 각 입력 컨트롤에 'numpad()'메소드를 한번 호출하면 클릭 이벤트와 'value passing'이 자동으로 이루어진다.

■ 그래프 예시

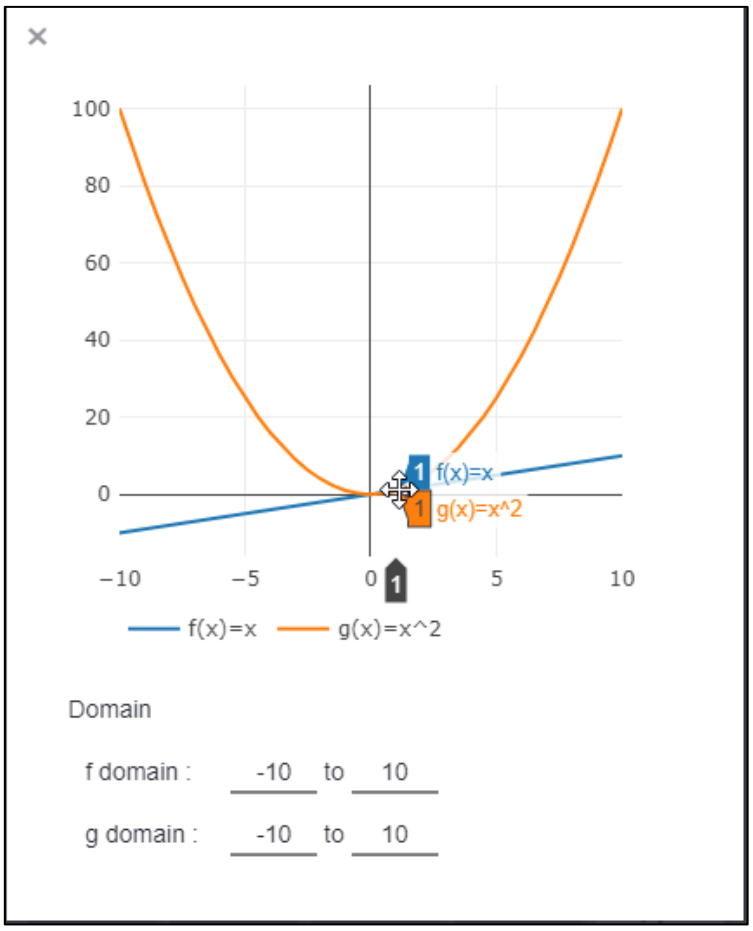
☐ f(x) = x, g(x) = x^2



☐ f(x) = sin(x), g(x) = cos(x)



- f(x) = x, g(x) = x^2의 교점 확인



- 그래프 위의 점에 커서를 올리면 x 값은 하단에 표기되고, y 값은 각 그래프마다 표기된다.

논의

- 구현 측면에서 성공적인 부분과 실패한 부분
  - 성공적인 부분
    - 과제에서 제시된 기능들을 모두 구현
  - 실패한 부분
    - 그래프의 정의역 구간이 좁아지면 그래프의 품질이 떨어지는데(곡선 표현) 이를 해결하지 못했다.
    - 그래프 위의 점을 나타내는 간격이 현재 0.5로 지정되어 있는데 0.1로 수정하면 값의 표현 방식이 부동소수점 표현 방식(0e-38)으로 나타나, 비교적 지저분하여 0.5로 고정하였다.
  - 어려웠던 부분
    - 인터넷 강의 'd3.js'로는 그래프를 표현하기에는 고려할 점이 너무 많아 오랜 시간 라이브러리를 찾아 해결하였다.
    - 사용자가 정의역 구간을 조정하는 인터페이스를 구현하기 위해 슬라이더, 이중 슬라이더, 삼중 슬라이더 등을 적용해보았으나 결국에 사용자가 입력하는 초기값을 추측하기에는 한계점이 있어, 텍스트 입력방식으로 방향을 바꾸었다.
    - 다양한 라이브러리 및 컨트롤을 찾아보고 해석하는 데에 시간이 꽤 걸렸다.

- 사용자 측면에서 긍정적인 측면과 부정적인 측면
  - 긍정적인 측면
    - 드래그로 손쉬운 원점 이동, 범례에 있는 그래프 표시 기능, 좌표 표시 등 다양한 기능들이 간단하면서도 직관적이게 표현되었다.
  - 부정적인 측면
    - 외부 라이브러리 및 컨트롤을 완벽하게 제어하지 못하여, 그래프의 품질 및 넘패드의 UI의 품질이 떨어졌다.
- 과제#3에 대한 전반적인 자체 평가 및 향후 개선 계획
  - 자체 평가
    - 외부 라이브러리 및 컨트롤의 소스코드를 활용하여 사용자와 상호작용을 최대한 이끌어내어 만족한다.
  - 향후 계선 계획
    - 전체적인 CSS 조정
- 유투브 시연 동영상 링크
  - https://youtu.be/pK21gU0SsMo