

휴먼컴퓨터인터페이스

#3

제출일 : 2018.05.27.(일)

담당교수 : 이강훈 교수님

학과 : 컴퓨터소프트웨어학과

학번 : 2014726049

이름 : 신숙



광운대학교
KwangWoon University

1. 개요

(1)기능적 요구조건 구현 완성도

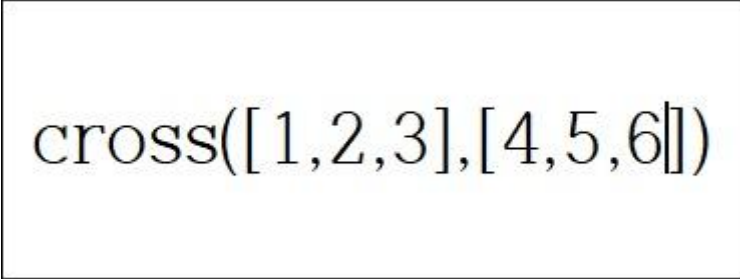
| 항목 | 여부 | 비고 |
|-----------------------------|----|------------------------|
| 1. 수식 입력 | | |
| (1) 정수, 실수, 복소수의 표현과 그 기본연산 | O | |
| (2) 벡터, 행렬의 표현과 그 기본 연산 | O | |
| (3) 자주 사용되는 상수 및 함수 지원 | O | |
| 2. 결과 출력 | | |
| (1) 올바른 입력 => 수식의 결과 값 | O | |
| (2) 잘못된 입력 => 오류 메시지 | O | |
| 3. 변수, 함수 및 정의 사용 | | |
| (1) 변수 : 최소 3개(예. x,y,z) | O | a, b, c, x, y, z 지원 |
| (2) 함수 : 최소 2개(예. f, g) | O | f(x), g(x), h(x) 함수 지원 |
| 4. 그래프 사용 | | |
| (1) 사용자 정의 함수에 대한 그래프 그리기 | O | 계산기에서 지정한 함수 표현 가능 |
| (2) 둘 이상의 함수에 대한 그래프 중첩 지원 | O | 3가지 함수 표현 |
| (3) 원점, 좌표축, 눈금 표시 | O | 좌표축으로 인한 원점과, 눈금 표시 |
| (4) 그래프 표시 영역 조절 가능 | O | 정의역 사용자 지정 및 확대, 축소 가능 |

(2)오픈소스 라이브러리 의존성

이번 과제에서는 그래프를 표현하기 위하여, math.js 뿐 아니라, plotly.js까지 사용하였다. 모든 수식은 math.js기반이고, 그래프는 math.js와 plotly.js에 의존한다.

(3)사용성 향상에 기여하는 핵심적인 상호작용 방식 요약

2번 과제에 이어서, 이번 과제에는 계산기의 복잡하고 불편한 사용, 익숙하지 않은 사용에 의거하여 '예시'기능을 추가하였다. 삼각함수들은 '삼각함수' 라는 모음을 클릭하면 바로 예시가 뜨고, 다른 함수들은 예를 들어 외적을 사용할 때, 'cross'를 클릭하면 예시가 뜨게 해두었다. 예시들은 한글에서 ctrl+n+m을 클릭하면 나타나는 수식 편집기를 이용하여 만들었고, 모든 이미지들의 크기를 통일하여 이질감이 없도록 하였다.



<- cross를 클릭하였을 때, 뜨는 이미지 원본.

그래프 기능은 math.js 와 연동하여 지원하는 plotly.js를 이용하여 사용하였다. plotly.js는 기본적으로 정의역 구간과 그로 인한 치역 범위를 정해주면, 자동으로 축이 생겨서 사용자가 알아보기 쉬운 인터페이스를 제공한다.

추가적으로, 그래프 뿐 아니라, 사용자가 지정한 함수들이 어떤 값을 가지고 있는 지 출력하도록 하였고, 사용자가 원하는 구역만 볼 수 있도록 정의역 구간을 정의할 수 있도록 만들어주었다.

또한 그래프들은 자기가 보고 싶은 함수들만 볼 수 있게, 그래프 옆 -f(x)를 클릭하면 출력되지 않거나 출력되게 상태를 변경할 수 있다.

2. 본문

(1) 기획 계획과 변경된 부분

이번 3차 과제에서는 2차 과제에서는 하지 못하였지만, 기본에 계획했던 예시들을 출력하게 할 수 있었고, jquery를 좀 더 익숙하게 사용할 수 있게 되어서, html에서 텍스트를 출력하는 것 뿐 아니라, jquery에서도 출력을 할 수 있게 만들어서, jquery 내부에서 함수들이 각각 무슨 값을 가지고 있는지 출력하도록 만들었다.

(2) 사용자 인터페이스의 구성 요소 및 사용 방법

먼저, Close키는 그래프, 정의역구간 지정, 함수에 대한 예시 이미지 등등 기본 외형에서 추가로 나타나는 것들을 종료하는 역할을 한다. 원치 않는 키를 누르거나, 다른 것을 사용해야 할 때, 누르는 역할이다.

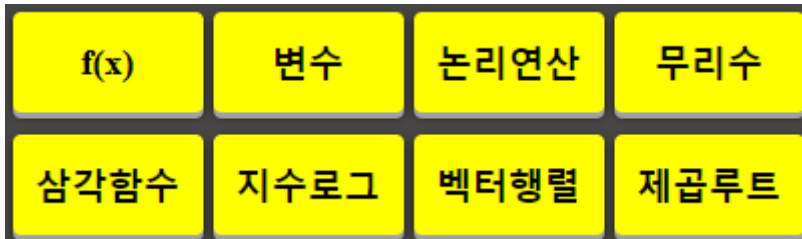
AC는 쓰던 수식을 0으로 초기화하는 역할을 한다. 한가지 계산이 끝났을 때, 다시 다른 식을 이용하기 위한 역할이다.

Backspace는 글자 1개를 지우는 역할이다. slice를 사용하여 마지막에 추가한 수를 지우거나, 함수들을 한가지씩 지우는 역할이다.

EV는 클릭 시 계산 혹은 변수와 함수에 값을 저장하는 역할을 하며, 예제 이미지들도 종료하는 역할을 한다.

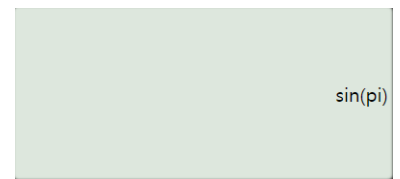
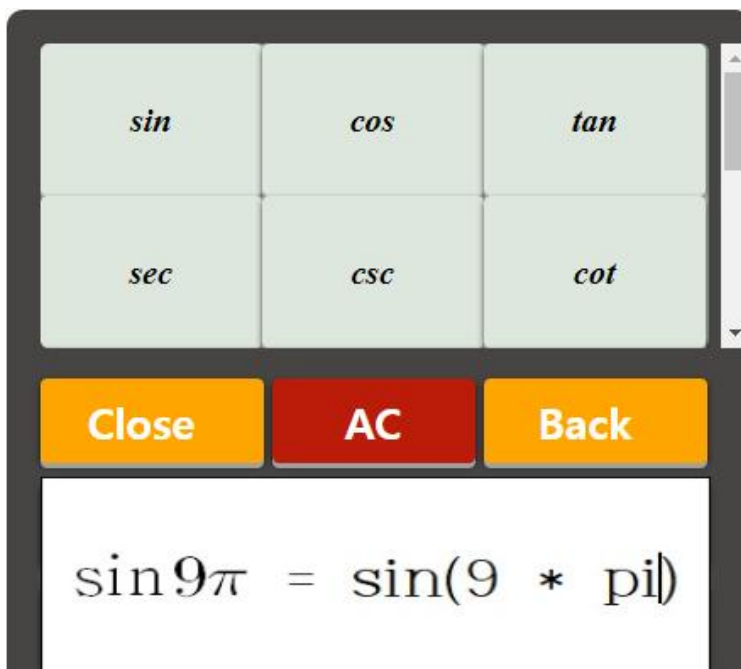
기본 수식들(*, /, +, - 등)의 사용방법은 보통의 계산기들과 동일하게 수와 수식(ex. $8 + 5$)을 누르고 EV키를 누르게 되면 계산이 되어 결과 값이 출력된다.

더 심화된 수식(삼각함수, 논리연산, 제곱, 루트, 벡터 등)을 이용하기 위해서는 계산기 중간에 있는 노란버튼들을 누르면 내부에 있는 기능들이 위에 출력된다.

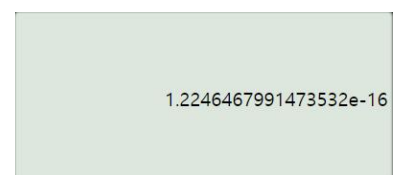


예로, 삼각함수라는 것을 클릭하게 되면, 상단에 아래 사진과 같은 선택지들이 scroll과 함께 뜨게 된다. 그리고 삼각함수는 다른 것들과는 다르게 똑같은 사용법을 가지고 있으나, 양이 매우 많기 때문에 모음만 클릭하면 예시들이 바로 출력되게 하였다. 그 이후로, sin을 클릭하고 예시와 같이 자기가 원하는 식을 치고 괄호를 닫으면 된다. 예시를 끄고 다른 함수를 사용 하고 싶으면, Close를 클릭하면 예시가 닫힌다. 다만, 사용할 함수목록들도 닫히게 되니 사용할 함수를 클릭 후, Close를 누르면 된다.

pi는 정확히 3.14가 아니라 무리수이기 때문에, 보통 생각하는 0이 아니라, 근삿값이 나오게 된다.

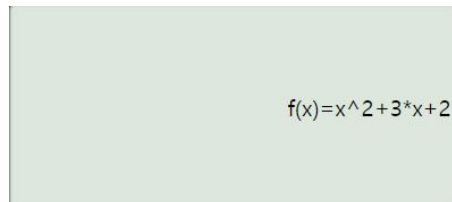


↓↓ EV 클릭



함수들은 노란 버튼들 중, f(x)를 누르게 되면, 'f(x), g(x), h(x), 그래프'가 뜨게 된다. 이 중 그래프를 제외하고는, 함수를 저장하는 역할을 하며, 사용방법은 익히 알다시피 (ex. $f(x) = x^2 + 3 * x + 2$) =을 이용하여 f(x)에 수식을 넣는 식

으로 사용하게 되며 저장은 EV를 누르면 저장이 된다. 그래프는 마지막 부분에서 설명하겠다.


$$f(x) = x^2 + 3x + 2$$

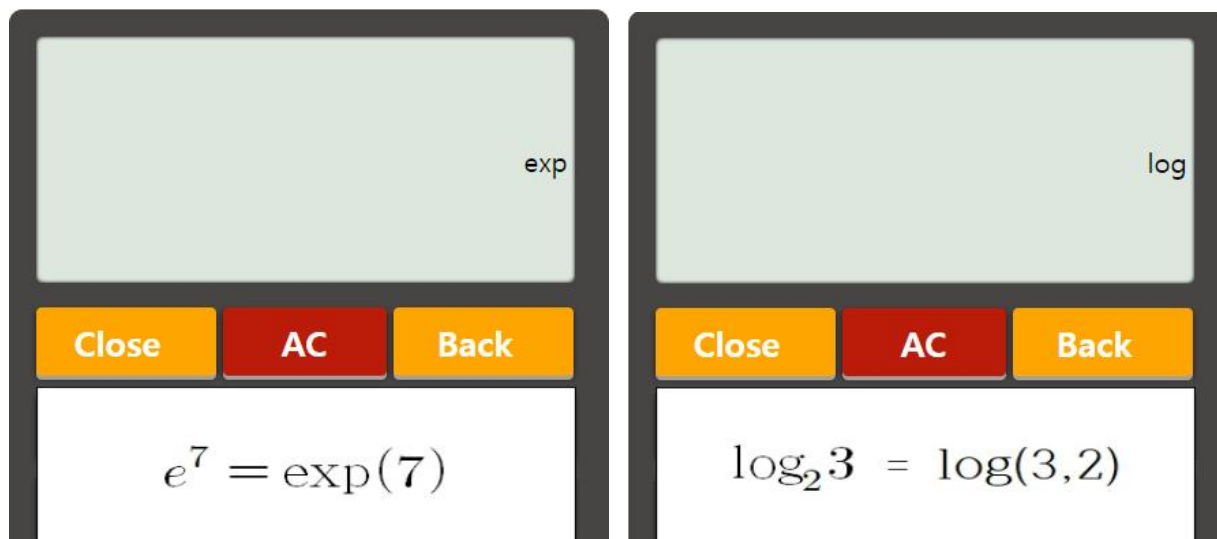
변수들은 위 함수와 같이 사용하거나, $x = 3$ 등 값을 넣을 수 있게 사용된다. 위의 함수에 이어서 $x = 3$ 을 누르고 EV를 누른 후, $f(x) \rightarrow$ EV를 누르면, 위 함수에 3이 들어가서 계산이 된다.



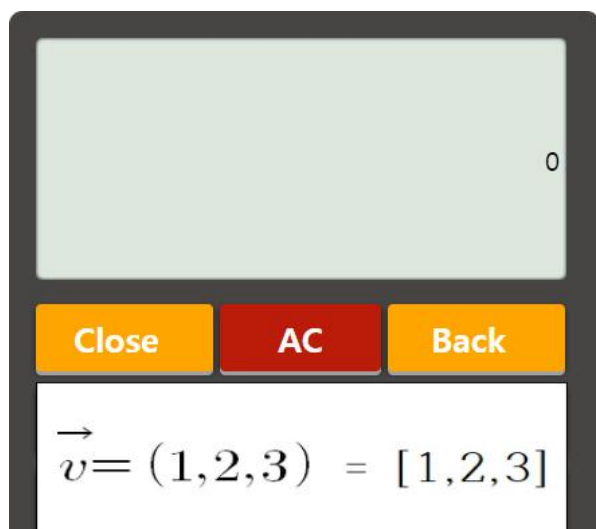
무리수 같은 경우는 pi, e, i 3가지를 상수처럼 사용할 수 있어서 그냥 수처럼 사용하면 된다.(ex. $\pi/2$)

논리 연산은 수식 혹은 값들이 같은지 다른지, 큰지, 작은지 등을 판별하는 함수로, 결과는 true 혹은 false가 나오게 된다. (ex. $7! = 7 \rightarrow \text{false}$)

지수로그에서 exp는 밑이 e인 지수함수로서 $\exp(10)$ 처럼 사용이 가능하며 이는 e^{10} 을 나타낸다. log는 $\log(8)$ 혹은 $\log(8,2)$ 2가지로 사용가능하며, 각각 $\log 8$ 과 $\log_2 8$ 을 나타낸다.



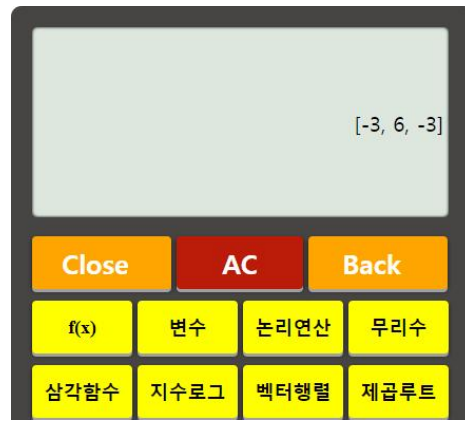
벡터는 기본 외형에 존재하는 [,] 인 대괄호를 사용하여 이용가능하며, 노란색의 함수 버튼을 이용하여 벡터를 클릭하게 되면, 값들이 나타나는 result 부분에는 변화가 없이 예시만 나타나게 된다.





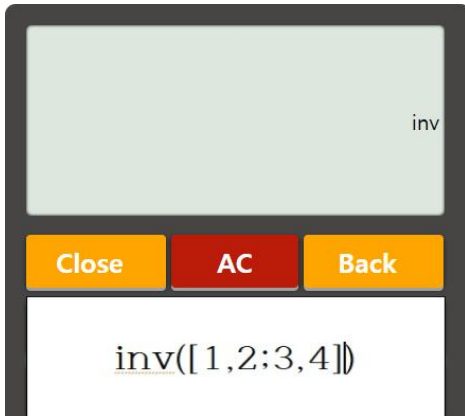
외적인 cross와 내적인 dot는 아래와 같이 예시가 뜨고, cross([1,2,3],[4,5,6]) 혹은 dot([1,2,3],[4,5,6]) 같이 입력을 해주고, EV를 누르면 계산이 된다.

EV 클릭

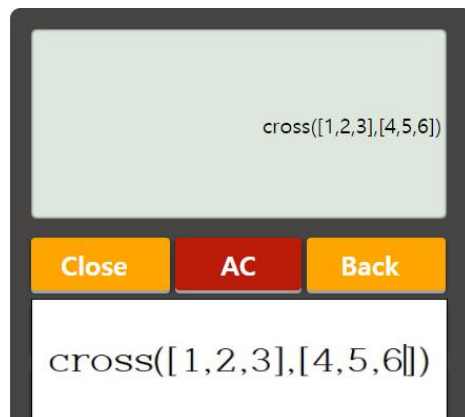
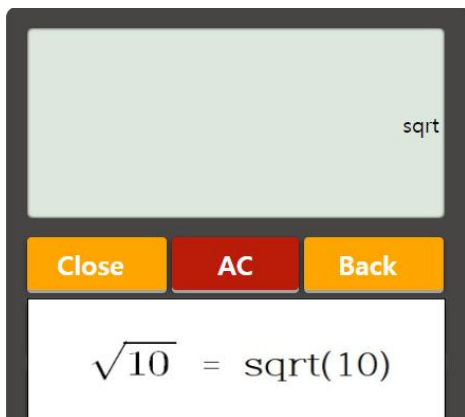


행렬도 역시 대괄호를 사용하지만, 내부에서 ;를 통해 구별한다. 2X2 행렬은[1,2;3,4] 와 같이 나타낸다. 행렬곱은 방식이 예제 이미지로 뜨지만, 위 결과 및 수식창에 뜨는 것은 * 가 끝이며, 그로 인해 먼저 행렬을 앞에 두지 않았다면, Backspace키를 이용하여, 앞 행렬을 먼저 쓴 후, *를 해주면 된다.

행렬식인 det와, 역행렬인 Inverse도 역시 사용방식이 뜨고 위 수식 창에 det와 inv를 각각 출력한다.



제곱루트를 클릭하게 되면, '^'와 'sqrt'가 뜨게 된다. sqrt는 sqaure root 의 줄임말로, 제곱근 밖에 구할 수 없다. ^인 제곱은 위 행렬곱과 마찬가지로 2^7 과 같이 사용해야 하지만, 입력은 ^밖에 되지 않으므로, Backspace 입력 후, 다시 수를 입력하고 ^을 입력해야한다.

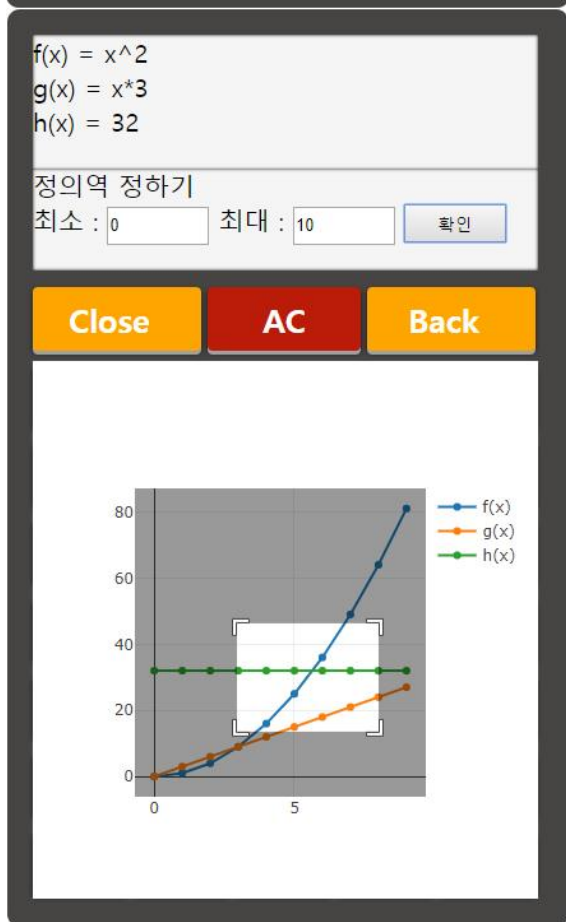
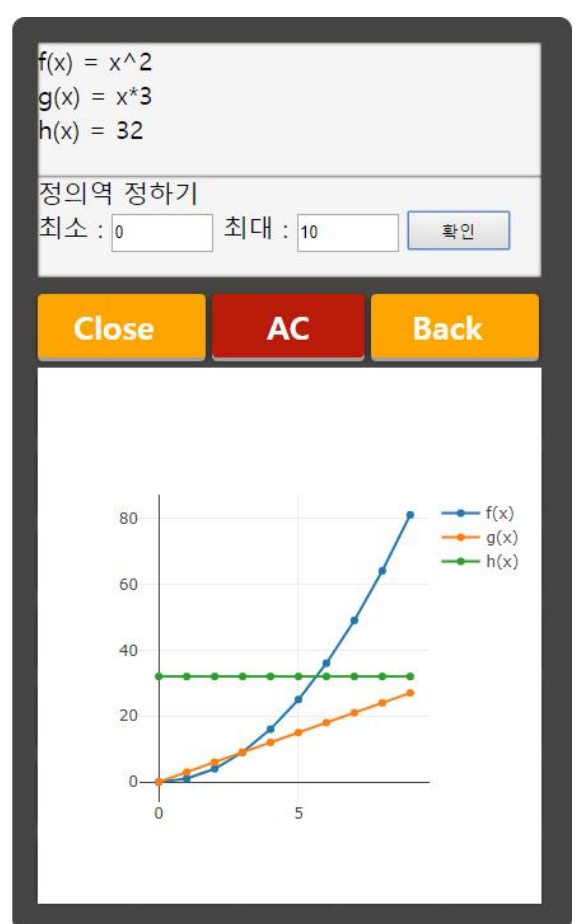


마지막으로 그래프기능을 설명하자면, 그래프는 노란버튼의 f(x)에 존재하며, 세 가지 함수(f(x),g(x),h(x))는 초기값이 0으로 설정되어 있다. 사용자가 임의로 f(x)에 식을 넣어주면 그 식이 기존 정의역인 -50~50까지의 구간에 나타내게 되고, 사용자가 다른 정의역을 원하면, result가 있던 대화창의 '정의역 구간 설정'에서 범위를 지정해주면 된다. *input type="number"에 pattern="[0,9]*"을 입력해두어서 만약 모바일에서 켜게 되면, 휴대폰 번호를 칠 때 키패드가 나타나게 된다.*

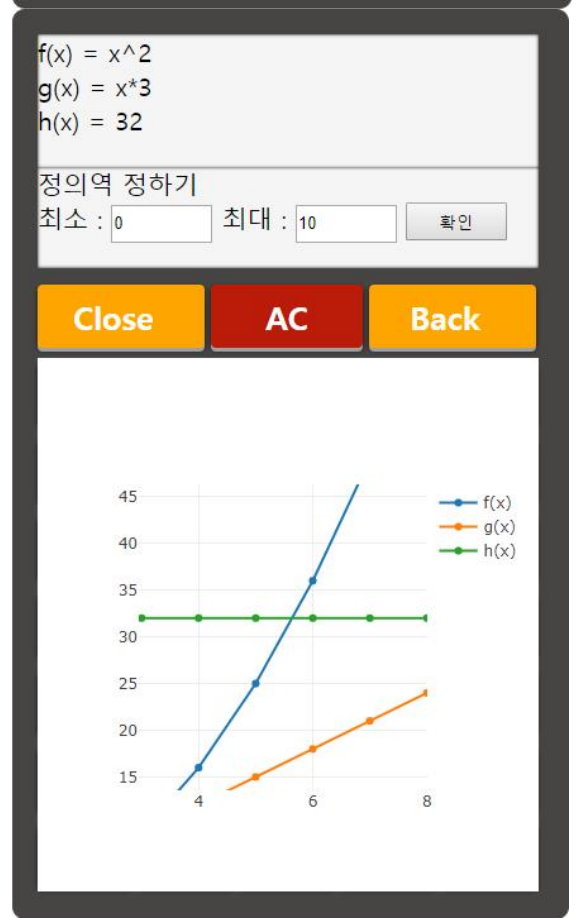
또한 f(x),g(x),h(x)가 각각 무슨 식이 들어있는지 알려주기 위해서 제일 위에 각각 함수들이 무슨 수식을 나타내는지 띄어주었다. 함수아래에 마우스를 누르고 있으면 확대 축소가 뜨며 또한, 그래프에서 보고 싶은 구간을 드래그하여 사각형을 만들면, 사각형구간이 크게 띄우게 된다.



정의역 구간
 변경 후
 확인버튼 클릭



보길 원하는
 구간 드래그



(3) 특징적인 상호작용 방식들에 대한 세부구현 방법

2번째 과제에 이어서 설명하자면, 이번에 나타내는 이미지들은, 하나의 div에 묶어서 모든 이미지들이 div 크기에 맞춰서 열려있고, 함수 버튼들을 누를 때마다 그 이미지를 띄워준다.

cross 버튼을 누를 때를 예로 들어서 아래에 보여주겠다.


```

<div class="selector">
  
  
  
  
  
  
  
  
  
  
  
</div>
.selector{
  width: 396px;
  height: 118px;
  position: absolute;
  display:none;
}
else if($(this).text() == 'cross'){
  displayValue += $(this).text();
  $('#result').text(displayValue);
  $('.selector').show();
  $('#idcross').show();
  Pre_img = '#idcross';
}

```

이미지 호출

div id인 selector의 css이다.

cross 버튼을 눌렀을 때, 작동하는 코드이다.
계산기 화면에 cross를 추가시키고, 이미지인 .selector를 출력하며 이미지인 #idcross를 출력시킨다.

그래프를 보여주는 코드는 plotly.js를 구글에 쳤을 때 나타나는 기본 'draw()' 함수코드에 3가지 함수를 모두 보여주기 위해서 추가적인 코드를 작성하였고, 정의역을 표현하기 위한 min, max 변수를 추가하였고, 그 위에서 f(x), g(x), h(x)에 수식을 저장하려고 EV를 누르면 slice를 통해 앞부분이 f(x), g(x), h(x)인지 판단하여 f, g, h 라는 변수에 수식을 저장한다.

또한, 정의역을 수정하고 확인 버튼을 눌렀을 때에도 draw()를 다시 해주어 수정한 범위를 보여준다.

<input type="number" pattern = "[0-9]*">에서 뒷부분은 원래 수를 변경할 때, 작은 화살표를 누르는데, 휴대폰에서는 너무 작을 것 같아서 구글에서 찾아보니, 뒤 코드를 입력하면 전화번호를 칠때와 같은 숫자 키패드가 뜬다고 한다.

```

if ($(this).text() == 'EV') {
  try {
    fullstring = displayValue;
    displayValue = parser.eval(displayValue).toString();
    var tokens = displayValue.split(' ');
    if (tokens[0] == 'function') {
      displayValue = tokens[0];
      if(fullstring.slice(0, 4) == 'f(x)'){
        f = fullstring.slice(5);
      }
      else if(fullstring.slice(0, 4) == 'g(x)'){
        g = fullstring.slice(5);
      }
      else if(fullstring.slice(0, 4) == 'h(x)'){
        h = fullstring.slice(5);
      }
    }
    $('#result').text(displayValue);
    displayValue = '0';
  }
}

```

EV를 눌렀을 때, slice를 이용하여 앞부분이 f(x), g(x), h(x)를 판단하여 각각 f, g, h에 수식을 저장시킨다.

```

function draw() {
  try {
    min = $('#min').val();
    max = $('#max').val();
    var expr1 = math.compile(f);
    var expr2 = math.compile(g);
    var expr3 = math.compile(h);

    var t1_xValues = math.range(min, max, 1).toArray();
    var t1_yValues = t1_xValues.map(function (x) {
      return expr1.eval({x: x});
    });
    var t2_xValues = math.range(min, max, 1).toArray();
    var t2_yValues = t2_xValues.map(function (x) {
      return expr2.eval({x: x});
    });
    var t3_xValues = math.range(min, max, 1).toArray();
    var t3_yValues = t3_xValues.map(function (x) {
      return expr3.eval({x: x});
    });

    var trace1 = {
      x: t1_xValues,
      y: t1_yValues,
      type: 'scatter',
      name: 'f(x)'
    };
    var trace2 = {
      x: t2_xValues,
      y: t2_yValues,
      type: 'scatter',
      name: 'g(x)'
    };
    var trace3 = {
      x: t3_xValues,
      y: t3_yValues,
      type: 'scatter',
      name: 'h(x)'
    };
    var data = [trace1, trace2, trace3];
    Plotly.newPlot('plot', data);
  } catch (err) {
    console.error(err);
    alert(err);
  }
}

<div id = "min-max">
  정의역 정하기<br>
  최소 : <input id="min" type="number" pattern="[0-9]*" value="-50"/>
  최대 : <input id="max" type="number" pattern="[0-9]*" value="50"/>
  <input id="btn" type="button" value="확인">
</div>
<div id="plot">
</div>
else if($(this).text() == '그래프'){
  $('#plot').show();
  draw();
  $('#function-value').show();
  $('#function-value').html("f(x) = " + f + "<br> g(x) = " + g + "<br> h(x) = " + h);
  $('#min-max').show();
}
$('#btn').click(function(){
  draw();
})

```

그래프를 그려주는 draw() 함수이다.
expr1, 2, 3는 각각 f, g, h 함수를
그리는 역할이다.

xValues들과 yValues들에서 정의역과
치역을 설정해 준다.

trace들은 그릴 data 리스트에 넣어주어
그리는 역할을 하게 된다.

Plotly.newPlot('plot',data);는
지정한 id가 'plot'인 div 영역에
그려주는 역할을 한다.

정의역을 설정하는 html 코드이다.

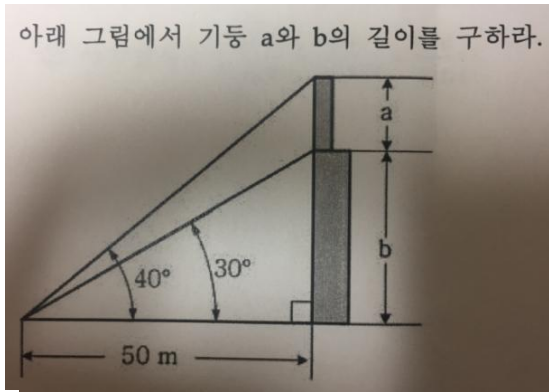
그래프가 그려질 id가 plot인 div이다.

‘그래프’ 버튼을 눌렀을 때 작동하는
코드로, 그래프와 함께 f(x), g(x), h(x)
들의 수식을 위 칸에 보여준다.

정의역을 설정한 후, 확인 버튼을 눌렀을 때 작동하는 코드이며,
지정한 정의역을 기준으로 다시 그려주는 역할을 한다.

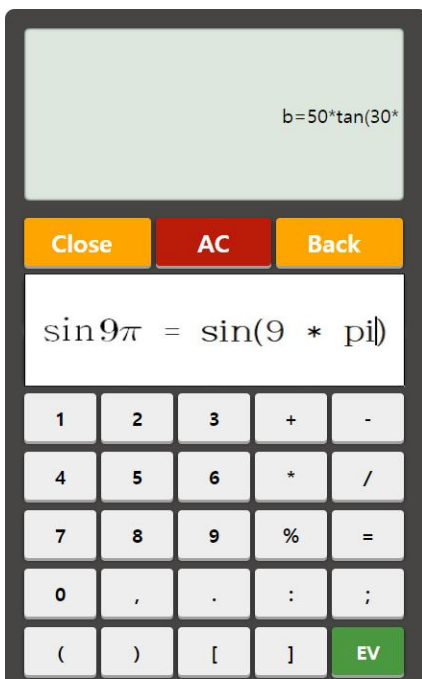
(3) 특징적인 상호작용 방식들에 대한 세부구현 방법

- (1)번 문제



위 문제는 삼각함수 공식을 사용해야 하는 문제이며, 밑변의 길이를 구하기 위해서 \tan 를 사용한다. 먼저 a 를 구하자면, $\tan(30 \times \pi / 180) = b / 50$ 을 이용해야 한다.

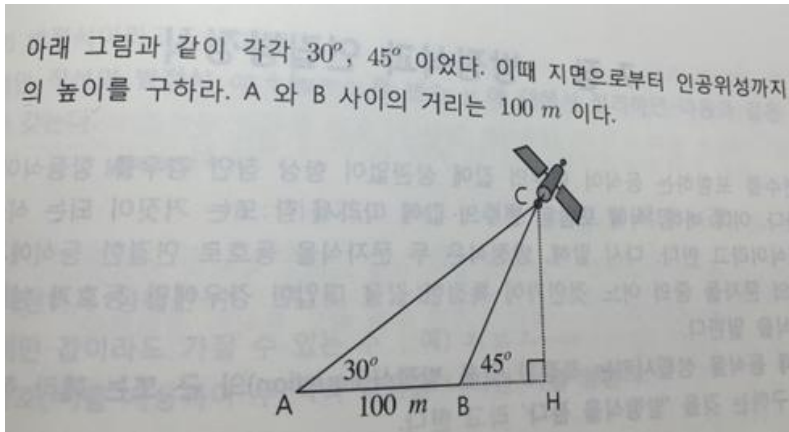
$b = 50 \times \tan(30 \times \pi / 180)$ 이므로 계산기를 이용해서 구하게 되면, 28.87이 나온다.



위에서 $a+b$ 를 x 로 두고, $x = 50 \times \tan(40 \times \pi / 180)$ 을 수행하면 41.95가 나오게 되며, b 가 28.87이니 이를 빼면 a 가 나오게 된다. $a = 13.1$, $b = 28.87$ 이 나온다.



- (2)번 문제



전체 AHC삼각형에서 HC거리를 h라고 두고, BH거리를 x라고 둔다.

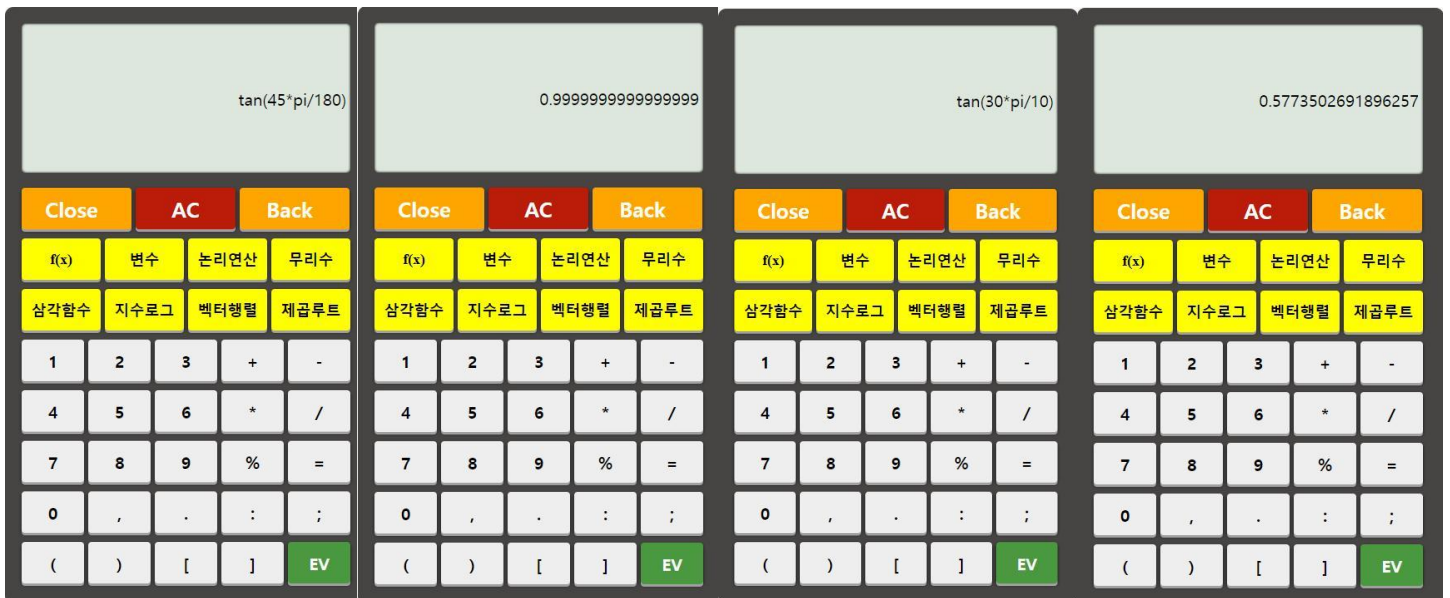
$\tan(30^\circ) = h/(100+x)$ 이다. h에 관해 정리를 하자면 $h=(100+x)\tan(30^\circ)$ 이다.

작은 BHC삼각형에서 $\tan(45^\circ) = h/x$ 이다. 이는 $h = x \cdot \tan(45^\circ)$ 이니

위 식과 연립해서 풀기 위해 $\tan(45^\circ)$ 를 먼저 계산하면, 1이 나온다. 따라서 $h = x$ 고,

위 중 $\tan(30^\circ)$ 를 계산하면, 0.58이 나온다. 따라서 $h=(100+h)*0.58$.

$0.42 \cdot h = 58 \Rightarrow h = 138.1$ 이 나온다.



(3)번 문제 : 두 벡터사이의 각도를 구하여라

A: $-2i + 4j$ B: $3i - 4j + 2k$ x : 두 벡터 사이의 각도

A = $[-2, 4, 0]$, B = $[3, -4, 2]$ 로 표현 가능하다.

두 벡터 사이의 각도는 $\cos(x) = \frac{AB\text{내적}}{(A\text{길이} * B\text{길이})}$

$$x = \frac{\text{dot}([-2, 4, 0], [3, -4, 2])}{(\text{sqrt}(20) * \text{sqrt}(29))}$$

Close AC Back

$\sqrt{10} = \text{sqrt}(10)$

| | | | | |
|---|---|---|---|----|
| 1 | 2 | 3 | + | - |
| 4 | 5 | 6 | * | / |
| 7 | 8 | 9 | % | = |
| 0 | , | . | : | ; |
| (|) | [|] | EV |

$$-0.9135002783911397$$

Close AC Back

f(x) 변수 논리연산 무리수

삼각함수 지수로그 벡터행렬 제곱루트

| | | | | |
|---|---|---|---|----|
| 1 | 2 | 3 | + | - |
| 4 | 5 | 6 | * | / |
| 7 | 8 | 9 | % | = |
| 0 | , | . | : | ; |
| (|) | [|] | EV |

$$\arccos(-0.913)$$

Close AC Back

$\sin 9\pi = \sin(9 * \pi)$

| | | | | |
|---|---|---|---|----|
| 1 | 2 | 3 | + | - |
| 4 | 5 | 6 | * | / |
| 7 | 8 | 9 | % | = |
| 0 | , | . | : | ; |
| (|) | [|] | EV |

acos로 $\arccos(\frac{AB\text{내적}}{(A\text{길이} * B\text{길이})})$ 을 넣으면, 2.7213이 나오는데 이는 라디안 값이 아니기 때문에, $(180/\pi)$ 를 곱해주면, $x = 156$ 도가 나온다.

$$2.7213 * (180/\pi)$$

Close AC Back

f(x) 변수 논리연산 무리수

삼각함수 지수로그 벡터행렬 제곱루트

| | | | | |
|---|---|---|---|----|
| 1 | 2 | 3 | + | - |
| 4 | 5 | 6 | * | / |
| 7 | 8 | 9 | % | = |
| 0 | , | . | : | ; |
| (|) | [|] | EV |

$$155.91900478895093$$

Close AC Back

f(x) 변수 논리연산 무리수

삼각함수 지수로그 벡터행렬 제곱루트

| | | | | |
|---|---|---|---|----|
| 1 | 2 | 3 | + | - |
| 4 | 5 | 6 | * | / |
| 7 | 8 | 9 | % | = |
| 0 | , | . | : | ; |
| (|) | [|] | EV |

3. 논의

(1) 구현 측면에서 성공적인 부분과 실패한 부분

이번 과제에서는 초기 아이디어를 모두 구현하여서 기분이 좋았다. 다만, 여러 가지 기능도 추가하고 싶었으나, 다른 키를 넣도록 배열을 바꾸지 못하였다. 초기 구상에서 노란버튼 8개만 생각하였기 때문에 다음버튼 목록, 이전버튼 목록을 넣을 공간을 생각해보았으나, 현재 배열에서 더 나은 배열을 찾지 못하였다. math.js에서 지원하는 최솟값, 최댓값 등 여러 가지를 더 만들고 싶었지만, 만들지 못하였고, 방식을 알았으니 더 나은 배열 혹은 크기를 더 키운다면, 앞으로 구현하는 것은 쉬울 것 같다.

초기부터 사용에 익숙하지 않은 사용자를 위하여 물건이 아니라 프로그램의 장점인 동적으로 나타내는 것을 통해 예시 혹은 도움말 기능을 추가하는 것을 생각해 왔지만, 2차 구현 때까지는 하지 못하였다. 하지만 이번 3차 구현때까지 하는 것을 목표로 해왔기 때문에 기본 밑작업은 어떻게 할 것인지를 다 생각해놓은 상태였고, 이미지 파일과 약간의 코딩으로 내가 밑그림을 그리던 그대로 구현을 할 수 있었다.

그리고 보고서를 쓰는 당일까지 정의역을 어떻게 사용자가 지정할지 생각을 되게 많이 했고, 응용소프트웨어실습 과목에서 하는 MessageBox를 생각하여 input type="number"를 떠올렸고, 막상 모바일에서 쓰기에는 수를 컨트롤하는 버튼이 너무 작아서 고민이었는데, 구글을 통해 pattern="[0-9]*"을 알아내게 되어 기분이 좋았다.

(2) 사용성 측면에서 긍정적인 측면과 부정적인 측면

이번 3차 구현을 통해, 사용자 측면에서 매우 긍정적이라고 생각한다. 만약 처음 사용하는 사람이고, 매뉴얼이 보기 힘들어서 보지 않더라도 누구나 어떤 함수가 어떤 기능을 하는지만 안다면 손쉽게 사용할 수 있기 때문이다. 함수를 어떤 식으로 사용하는지 모르더라도, 사용 예시 이미지가 뜨기 때문에 방식만 따라하더라도 매우 쉽게 계산기를 사용가능할 것이다.

하지만 여전히 많은 기능이 있지는 않기 때문에 단순 수학문제만을 푸는 게 아니라, 1차 과제 때, 인터뷰를 하였던 전자회로 쪽 문제를 푸는 사람들에게는 전혀 쓸 수 없는 계산기일 것이다.

2차 과제 시연영상 : <https://youtu.be/VOISwC16hSI>

3차 과제 시연영상 : <https://youtu.be/Oi6XSWDLzcs>