

휴먼 컴퓨터 인터페이스

과제 #3. 직접 조작 계산기 구현

제출일	2021.06.12
소속	소프트웨어학부
학번	2016726022
이름	이지원

목차

I 개요

- 1 인터페이스 요구조건에 대한 구현 완성도 요약 (표)
- 2 오픈소스 라이브러리 의존성 요약

II 본문

- 1 설계
- 2 구현
- 3 평가

III 논의

- 1 구현 측면에서 성공적인 부분과 실패한 부분
- 2 사용성 부분에서 긍정적인 부분과 부정적인 측면
- 3 과제 결과에 대한 전반적인 자체 평가 및 향후 개선 계획

I - 1 인터페이스 요구조건에 대한 구현 완성도 요약 (표)

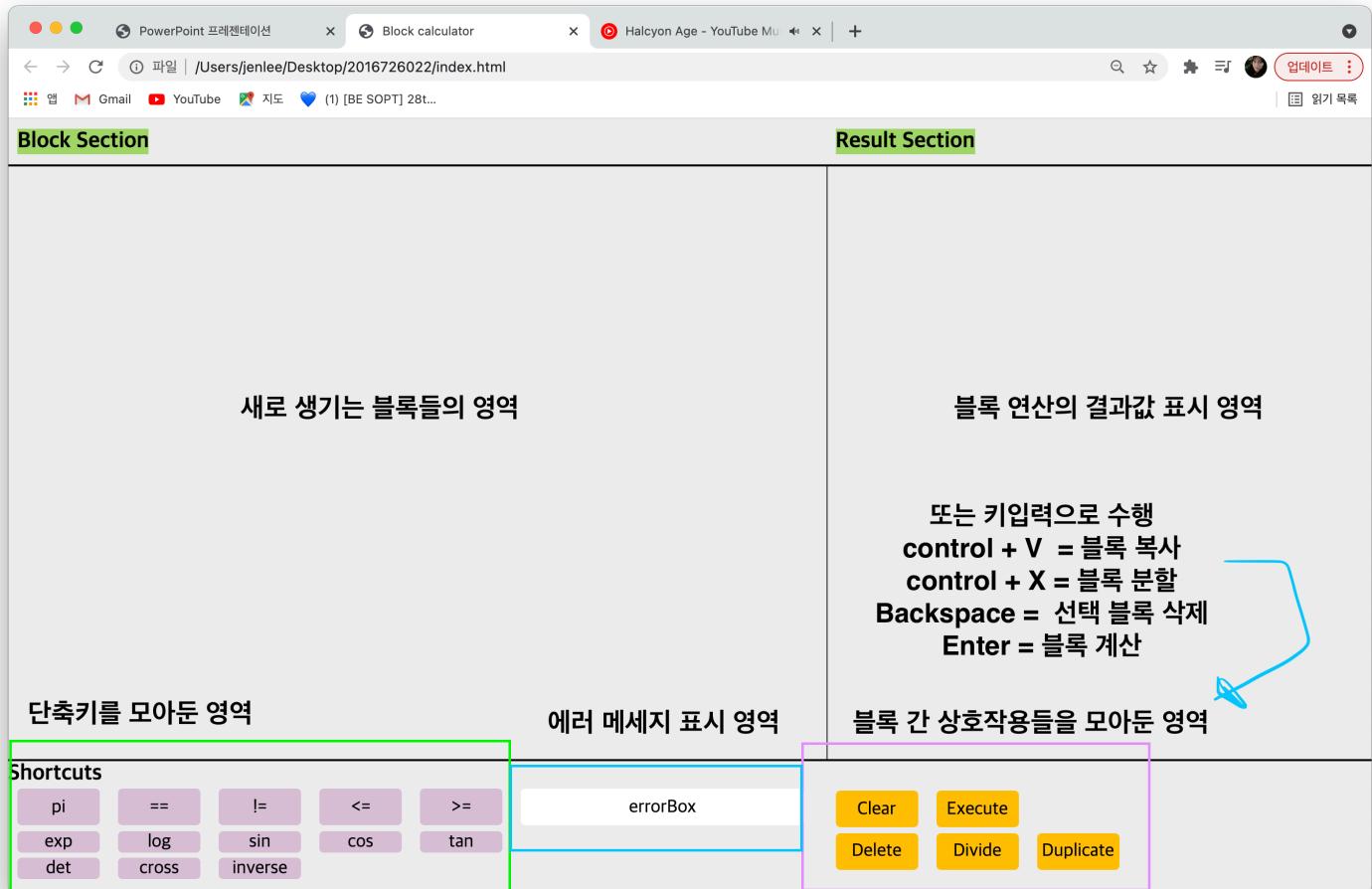
요구조건	수행 여부
기능적 요구 조건	
수식 입력	O
벡터, 행렬의 표현과 그 기본 연산	O
자주 사용되는 상수 및 함수 지원	O
결과 출력	O
변수, 함수 정의 및 사용	O
제약 조건	
클라이언트 측 스크립트만 사용	O
구글 크롬 웹 브라우저 호환 필수	O
인터페이스 요구 조건	
수, 연산자, 변수, 함수, 수식	O
대화형 상호작용 (생성, 이동, 조립, 실행, 분해, 복제, 소멸)	O
예시 인터페이스와의 차별성	O

I - 2 오픈 소스 라이브러리 의존성 요약

```
<script src='https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js' type='text/javascript'></script>
<script src='https://cdnjs.cloudflare.com/ajax/libs/fabric.js/2.4.6/fabric.min.js' type="text/javascript"></script>
<script src="resources/math.js" type ="text/javascript"></script>
<script src="resources/index.js" type="text/javascript"></script>
```

Fabric 과 math js 를 제외한 외부 라이브러리를 사용하지는 않았다.

II - 1 설계



전체적인 인터페이스를 요약

삼각함수나 행렬, 비교 부호 같은 수학적 요소들의 입력을 쉽게 하기 위해 하단에 단축키 버튼을 만들었다. 또한 상호작용, 에러메세지를 표시하고, 버튼이 눌리거나 미리 지정해둔 단축키(컨트롤 + V)를 입력하면 블록 간 상호작용을 통해 결과를 보여준다. 또한 블록들은 Block Section 무작위 위치에 생성되고, 계산 결과나 상호작용 결과는 Result Section에 생성되어 블록이 많을 경우 헷갈리지 않게 설계하였다.

```

<div class="divider"></div>
<div class="shortcut">
  <div class="shortcut__column">
    <h2 class="shortcut__header">Shortcuts</h2>
    <div class="shortcut__row">
      <span class="shortcut__btn" onclick="{MathApp.shortcutPress('pi')}">pi</span>
      <span class="shortcut__btn" onclick="{MathApp.shortcutPress('==')}">==</span>
      <span class="shortcut__btn" onclick="{MathApp.shortcutPress('!=')}">!!=</span>
      <span class="shortcut__btn" onclick="{MathApp.shortcutPress('<=')}"><=</span>
      <span class="shortcut__btn" onclick="{MathApp.shortcutPress('>=')}">>=</span>
    </div>
    <div class="shortcut__row">
      <span class="shortcut__btn" onclick="{MathApp.shortcutPress('exp')}">exp</span>
      <span class="shortcut__btn" onclick="{MathApp.shortcutPress('log')}">log</span>
      <span class="shortcut__btn" onclick="{MathApp.shortcutPress('sin')}">sin</span>
      <span class="shortcut__btn" onclick="{MathApp.shortcutPress('cos')}">cos</span>
      <span class="shortcut__btn" onclick="{MathApp.shortcutPress('tan')}">tan</span>
    </div>
    <div class="shortcut__row">
      <span class="shortcut__btn" onclick="{MathApp.shortcutPress('det')}">det</span>
      <span class="shortcut__btn" onclick="{MathApp.shortcutPress('cross')}">cross</span>
      <span class="shortcut__btn" onclick="{MathApp.shortcutPress('inverse')}">inverse</span>
    </div>
  </div>
  <div class="shortcut__column">
    <div class="shortcut__row" id="cal_row">
      <span class="cal_btn" onclick="{MathApp.calPress('Clear')}">Clear</span>
      <span class="cal_btn" onclick="{MathApp.calPress('Execute')}">Execute</span>
    </div>
    <div class="shortcut__row">
      <span class="cal_btn" onclick="{MathApp.calPress('Delete')}">Delete</span>
      <span class="cal_btn" onclick="{MathApp.calPress('Disassemble')}">Divide</span>
      <span class="cal_btn" onclick="{MathApp.calPress('Duplicate')}">Duplicate</span>
    </div>
  </div>
</div>
</body>
</html>

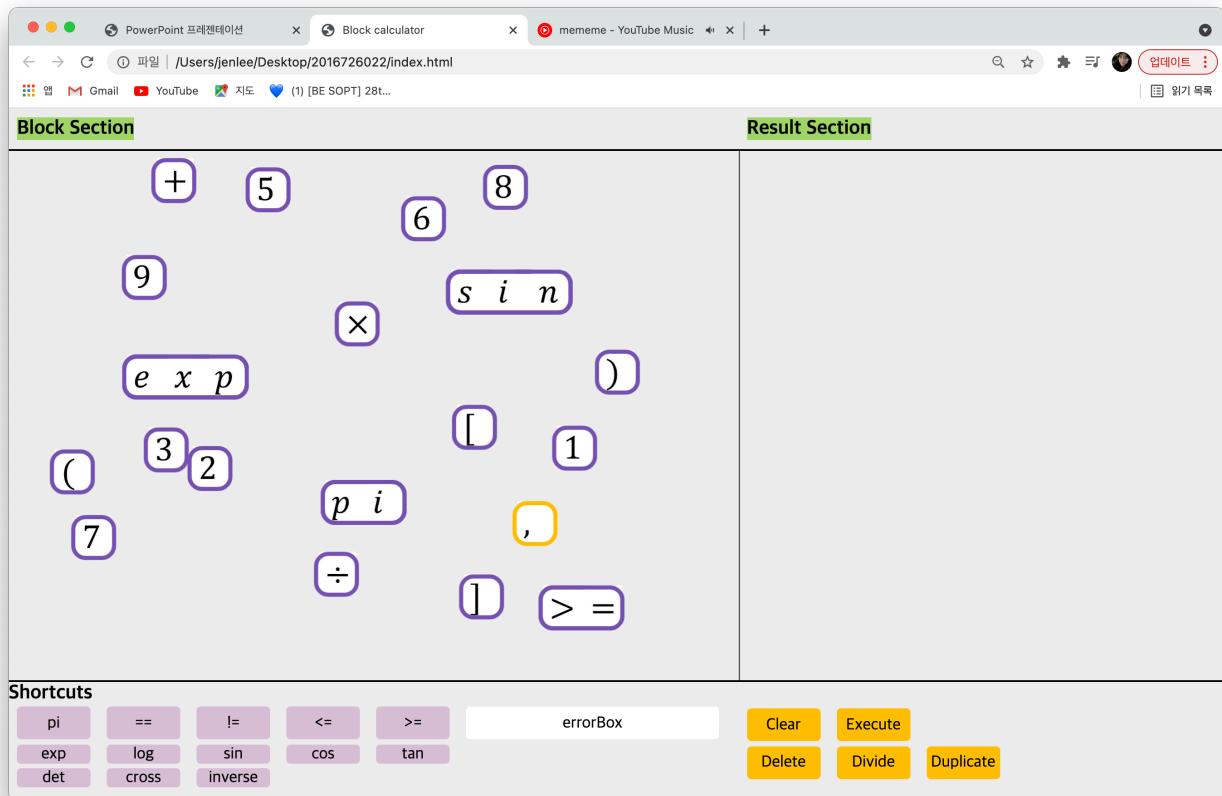
```

단축키 생성

HTML 을 사용하여 키보드 입력을 통한 상호작용 뿐만이 아니라 직접 입력하기 번거로울 수 있는 삼각함수들이나 부호들을 바로 입력할 수 있는 단축키들을 생성하고 있다. 이들은 직접 작성한 스크립트 문서 안에서 if else 문으로 분기되어 각각의 기능에 맞는 함수를 실행하며 처리된다.

또한 블록들을 한꺼번에 지울 수 있는 Clear 버튼, 결과를 얻을 수 있는 실행버튼, 블록 copy 기능, 블록을 나눌 수 있는 Divide 등등은 미리 지정해둔 단축키 (control + v) 등을 통해 실행될 수 있지만, 버튼을 통해서도 가능하다.

1. 생성



직접 키를 입력하거나, 아래 단축키를 클릭하여 쉽게 수학적 연산 요소들을 추가 가능하다.

2. 이동

각각의 블록들을 이동시킬 수 있다.

3. 조립

The screenshot shows the 'Block calculator' application interface. It has two main sections: 'Block Section' on the left and 'Result Section' on the right. In the 'Block Section', there are three blocks:

- A yellow block containing the text `c o s (p i ÷ 2) =`
- A purple block containing the text `4 + 5 ! = 5`
- A purple block containing the text `[3 , 9 , 6] + [-1 , -2 , -3] =`

In the 'Result Section', the output is currently blank.

At the bottom, there is a 'Shortcuts' panel with various buttons for mathematical functions like pi, exp, det, ==, log, cross, !=, sin, inverse, <=, cos, >=, tan, and errorBox. There are also buttons for Clear, Execute, Delete, Divide, and Duplicate.

블록을 오른쪽이던 왼쪽이던 가깝게 위치시키면 블록들이 조립된다.

4. 실행

The screenshot shows the 'Block calculator' application interface after the blocks have been assembled. The 'Block Section' now contains a single large block:

$$c \circ s (p \circ i \div 2) = 4 + 5 ! = 5 [3 , 9 , 6] + [-1 , -2 , -3]$$

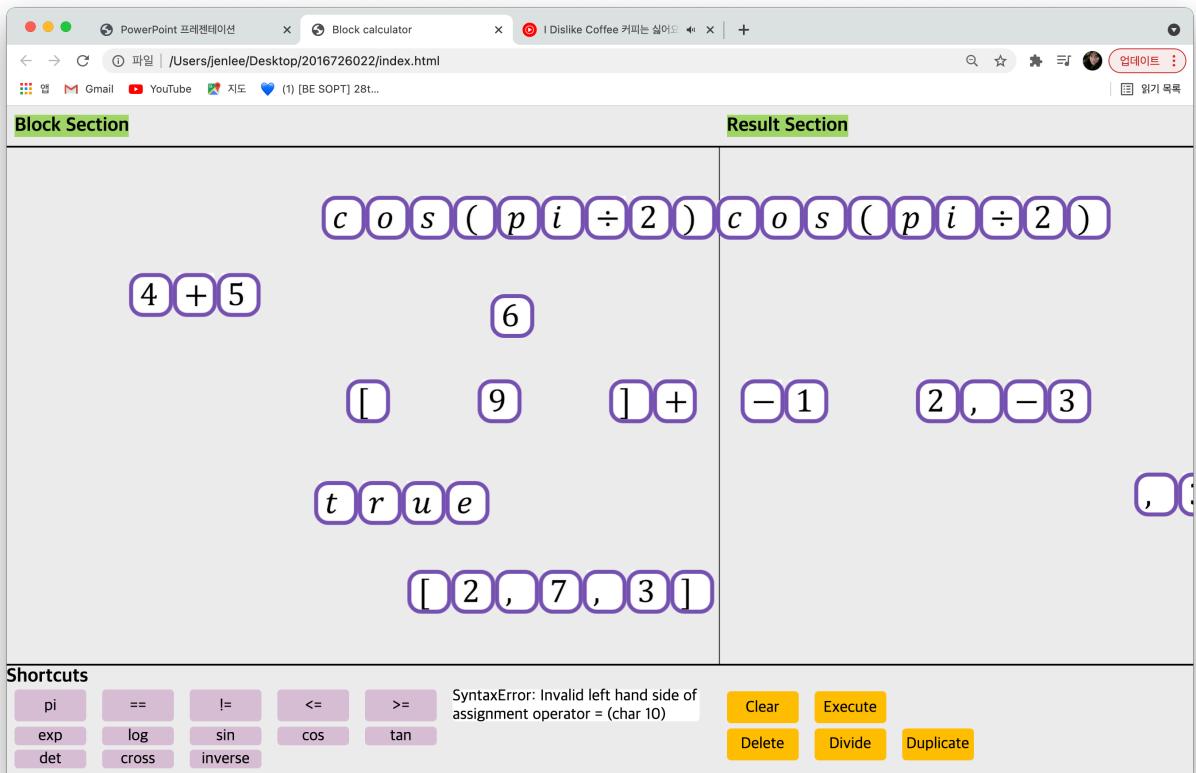
The 'Result Section' on the right displays the results of the execution:

- The first part of the block, `c o s (p i ÷ 2)`, resulted in `6 . 1 2 3 2 3 3 9 9 5`.
- The second part, `4 + 5 ! = 5`, resulted in `t r u e`.
- The third part, `[3 , 9 , 6] + [-1 , -2 , -3]`, resulted in `[2 , 7 , 3]`.

At the bottom, there is a 'Shortcuts' panel with various buttons for mathematical functions like pi, exp, det, ==, log, cross, !=, sin, inverse, <=, cos, >=, tan, and errorBox. There are also buttons for Clear, Execute, Delete, Divide, and Duplicate. A message in the status bar indicates a SyntaxError: Invalid left hand side of assignment operator = (char 10).

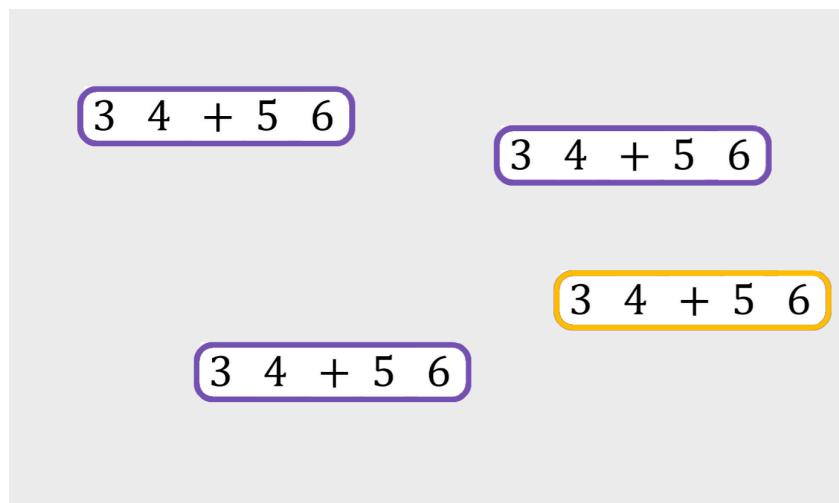
오른쪽 하단의 Execute 버튼을 클릭하거나, 엔터를 누르면 계산결과 블록이 Result Section에 위치하게 된다.

5. 분해



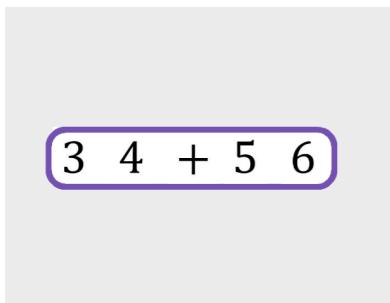
오른쪽 하단의 Divide 버튼을 클릭하거나, control X 를 누르면 블록이 쪼개진다.

6. 복제



오른쪽 하단의 Duplicate 버튼을 클릭하거나, control V 를 누르면 블록이 복제된다.

7. 소멸



오른쪽 하단의 Delete 버튼을 클릭하거나 모든 블록을 다 지우고 싶다면 Clear 버튼, Backspace 를 누르면 블록이 소멸된다.

II - 2 구현

1. 생성

```
> MathApp.makeASymbol = function(name, position) {
>   let size = {
>     width: SYMBOL_WIDTH,
>     height: SYMBOL_HEIGHT
>   };
>
>   new MathApp.Symbol(position, size, name);
> }
```

기본적으로 블록의 생성은 Symbol 객체를 추가함으로서 구현되었다. 전체 코드에서 새로운 블럭이 생성되는 경우가 많아 이를 makeASymbol 함수로 정의하여 구현하였다.

초기에 숫자나 키보드의 키를 눌렀을 때 블록이 추가되는 부분은

```
MathApp.handleKeyPress = function(key) {
  if (key in this.symbol_paths)
  {
    let size = {
      width : SYMBOL_WIDTH,
      height : SYMBOL_HEIGHT
    };

    let position = {
      x : (Math.random() * (this.canvas.width-size.width)
            + size.width / 2) * (SECTION_RATE),
      y : (Math.random() * (this.canvas.height-size.height)
            + size.height / 2) - (window.innerHeight * (1.0 / 100.0))
    };
    new MathApp.Symbol(position, size, key);
  }
}
```

handleKeyPress 함수에서 처리한다. Key 를 누르면 이벤트를 받아 사이즈와 포지션을 정의하고, 새로운 Symbol 객체를 생성한다. Position 의 x 좌표를 계산해 block_section 내 랜덤한 위치에서만 생성되도록 하였다.

```
MathApp.canvas.add(background);
block.visual_items.push(background);

for(let i=0; i < name.length; i++){
  if(name[i] in MathApp.symbol_paths){
    let path = "resources/" + MathApp.symbol_paths[name[i]] + ".jpg";
    fabric.Image.fromURL(path, function(img) {

      img.scaleToWidth(SYMBOL_WIDTH);
      img.scaleToHeight(SYMBOL_HEIGHT);
      let img_w = img.getScaledWidth();
      let img_h = img.getScaledHeight();

      img.set({
        left: position.x - size.width / 2 + SYMBOL_WIDTH * i + 6,
        top: position.y - img_h / 2,
        selectable: false
      });

      let boundary = new fabric.Rect({
        left: position.x - size.width/2,
        top: position.y - size.height/2,
        width: size.width,
        height: size.height,
        fill: "rgba(0,0,0,0)",
        stroke: UNSELECTED_COLOR,
        strokeLineJoin: 'round',
        rx: 15,
        ry: 15,
        strokeWidth: 5,
        selectable: false
      });

      MathApp.canvas.add(img);
      MathApp.canvas.add(boundary);

      block.visual_items.push(img);
      block.visual_items.push(boundary);

    });
  }
}
```

Symbol 객체 내에서 백그라운드를 정의하고, 해당하는 키에 맞는 이미지를 불러와 세팅한다.

2. 이동

이동은 handleMouseDown, handleMouseMove, handleMouseUp에 의해 조작된다.

```
MathApp.handleMouseDown = function(window_p) {
    if(MathApp.isInCanvas(window_p))
    {
        let canvas_p = MathApp.transformToCanvasCoords(window_p);

        if( MathApp.selected_block != null )
        {
            MathApp.selected_block.onDeselected();
            MathApp.selected_block = null;
        }
        let block = MathApp.findBlockOn(canvas_p);
        if(block != null)
        {
            MathApp.selected_block = block;
            MathApp.selected_block.onSelected();
        }

        MathApp.is_mouse_dragging = true;
        MathApp.mouse_drag_prev = canvas_p;
        MathApp.canvas.requestRenderAll();
    }
    else
    {
        MathApp.is_mouse_dragging = false;
        MathApp.mouse_drag_prev = {x:0, y:0};
    }
}
```

```
MathApp.handleMouseMove = function(window_p) {
    let canvas_coord = MathApp.transformToCanvasCoords(window_p);
    if(MathApp.is_mouse_dragging)
    {
        if(MathApp.selected_block != null){
            let tx = canvas_coord.x - MathApp.mouse_drag_prev.x;
            let ty = canvas_coord.y - MathApp.mouse_drag_prev.y;
            MathApp.selected_block.translate({x: tx, y: ty});
        }
        MathApp.mouse_drag_prev = canvas_coord;
        MathApp.canvas.requestRenderAll();
    }
}
```

```

MathApp.handleMouseUp = function(window_p) {

    const canvas_coord = MathApp.transformToCanvasCoords(window_p);
    const target = MathApp.findBlockOn(canvas_coord);

    if(target != null){
        const block = MathApp.is_overlapped();
        if(block != undefined)
            MathApp.assemble(block);
    }

    if(MathApp.is_mouse_dragging){
        MathApp.is_mouse_dragging = false;
        MathApp.mouse_drag_prev = {x:0, y:0};
    }
    MathApp.canvas.requestRenderAll();
}

```

이동뿐 만이 아니라 다음 항목인 조립의 기능을 구현하는 부분인 handleMouseUp 함수이다. 현재 위치의 블록을 찾아서 만약 겹친다면 이를 합친다.

3. 조립

```

MathApp.is_overlapped = function(){
    const selectedBlock = MathApp.selected_block;
    let overlapedBlock = undefined;

    if(selectedBlock != null)
    {
        MathApp.blocks.forEach(block => {
            const axisX = Math.abs(selectedBlock.position.x - block.position.x);
            const axisY = Math.abs(selectedBlock.position.y - block.position.y);

            if((axisX == 0) && (axisY == 0))
            {
                return;
            }

            else if((axisX <= (selectedBlock.size.width / 2) + (block.size.width / 2)) &&
            (axisY <= (selectedBlock.size.height) + (block.size.height / 2)))
            {
                MathApp.canvas.requestRenderAll();
                overlapedBlock = block;
            }
        })
        return overlapedBlock;
    }
}

```

두 객체의 총돌 여부를 확인하고, 겹친다면 이들을 합친다.

```
MathApp.assemble = function(target) {
    const selected_block = MathApp.selected_block;

    let size = {
        width: target.size.width + selected_block.size.width,
        height: target.size.height
    };

    let position = {
        x: target.position.x,
        y: target.position.y
    };

    if( target.position.x <= selected_block.position.x )
    {
        new MathApp.Symbol(position, size, target.name + selected_block.name);
    }

    else
    {
        new MathApp.Symbol(position, size, selected_block.name + target.name);
    }

    selected_block.destroy();
    target.destroy();
    MathApp.canvas.bringToFront(item);

    MathApp.selected_block = null;
}
```

여기서 왼쪽에 붙이면 왼쪽에 결합되고, 오른쪽에 붙이면 오른쪽에 결합된다.

4. 실행

```

MathApp.handleKeyDown = function(key) {
  if( MathApp.selected_block != null )
  {
    if(key == '')
    {
      MathApp.selected_block.destroy();
      MathApp.selected_block = null;
      return;
    }

    else if( isCtrl == true && key == 'X' )
    {
      MathApp.divide();
      MathApp.selected_block = null;
      return;
    }

    else if( isCtrl == true && key== 'V' )
    {
      MathApp.duplicate();
    }

    if( key == "enter" )
    {
      MathApp.evaluate(MathApp.selected_block);
    }
  }
}

```

handleKeyDown 함수이다. 각각의 단축키에 대응하는 함수들을 보여주고 있다.
각각 분해, 복제, 계산 등의 기능을 한다.

```

MathApp.evaluate = function(block) {
  let input = block.name
  let result = ""

  try {
    result = parser.eval(input).toString()
    let tokens = result.split(' ');

    if ( tokens[0] == 'function' ){
      result = tokens[0];
    } else {
      result = result.replace(/(\s*)/g, '');
    }

    let position = [ x: SYMBOL_MARGIN + window.innerWidth * (SECTION_RATE)
    + (result.length * (SYMBOL_WIDTH / 2)), y: block.position.y ];
    let size = { width: SYMBOL_WIDTH * (result.length), height: SYMBOL_HEIGHT};
    new MathApp.Symbol(position, size, result);
  }

  catch(e) {
    var cur = document.getElementById("errorBox");
    cur.textContent = e
  }
}

```

위 함수는 식들을 계산하는 함수이다. 계산의 결과를 Result block에 들어가도록 해준다.

5. 분해

```
MathApp.divide = function() {
  if( MathApp.selected_block == null || MathApp.selected_block == undefined ) {
    return;
  }

  let selectedBlock = MathApp.selected_block;

  for(var i = 0; i < selectedBlock.name.length; i++){
    let position = {
      x: selectedBlock.position.x + 55 * i,
      y: selectedBlock.position.y
    };

    this.makeASymbol(selectedBlock.name[i], position)
  }

  selectedBlock.destroy();
  MathApp.selected_block = null;
}
```

정해진 입력이 들어오면 makeASymbol 을 이용해 각각의 블록으로 분해하는 모습을 볼 수 있다.

6. 복제

```
MathApp.duplicate = function() {  
  
    let block = MathApp.selected_block;  
    let size = {  
        width: block.size.width,  
        height: block.size.height  
    }  
    let position = { x: block.position.x + 30, y: block.position.y + 30};  
    new MathApp.Symbol(position, size, block.name);  
  
}
```

각각 x 좌표과 y좌표에 30을 더한 위치에 선택된 블록과 똑같은 블록을 생성한다.

7. 소멸

```
MathApp.Block.prototype.destroy = function() {  
    if(this == MathApp.selected_block)  
    {  
        MathApp.selected_block = null;  
        this.onDeselected();  
    }  
  
    this.visual_items.forEach(item => {  
        MathApp.canvas.remove(item);  
    });  
    this.visual_items = [];  
  
    let index = MathApp.blocks.indexOf(this);  
    if(index > -1)  
    {  
        MathApp.blocks.splice(index, 1);  
    }  
}
```

선택된 블록을 소멸시킨다.

8. 잘못된 입력에 대한 오류 메세지

3 4 +

SyntaxError: Unexpected end of expression (char 4)

```
catch(e) {
    var cur = document.getElementById("errorBox");
    cur.textContent = e
}
```

다 완성되지 못한 식을 선택하고, 엔터를 누르자(계산하기) 오른쪽 값이 없는 수식이 되었으므로 에러메세지를 출력하는 것을 볼 수 있다.

```
MathApp.calPress = function(keyword) {
    if(keyword == 'Clear') {
        for(let i = 0; i < this.blocks.length; i++)
        {
            let block = this.blocks[i];
            setTimeout(function() { MathApp.handleKeyDown = function(key) {
                if( MathApp.selected_block != null )
                {
                    if(key == ' ')
                    {
                        MathApp.selected_block.destroy();
                        MathApp.selected_block = null;
                        return;
                    }
                    else if( isCtrl == true && key == 'X' )
                    {
                        MathApp.divide();
                        MathApp.selected_block = null;
                        return;
                    }
                    else if( isCtrl == true && key== 'V' )
                    {
                        MathApp.duplicate();
                    }
                    if( key == "enter" )
                    {
                        MathApp.evaluate(MathApp.selected_block);
                    }
                }
            }}
```

또한 위의 코드들을 통해 분기를 처리하고 단축 키 처리에 대한 로직을 수행하는 것을 볼 수 있다.

예시 인터페이스와의 차별성

Shortcuts

pi

==

!=

<=

>=

exp

log

sin

cos

tan

det

cross

inverse

S
€

Clear

Execute

Delete

Divide

Duplicate

1. 각종 단축키 버튼과 키보드 단축키 지원

시각적으로 보여지는 단축키들 뿐만이 아니라,

부분 삭제 Backspace

블록 복제 control + V

블록 분할 control + X

블록 실행 Enter

같은 키보드 단축키도 지원하여 편리성을 증대시키고자 하였다.

2. 구역의 분리

Block Section	Result Section

블록들은 Block Section 무작위 위치에 생성되고, 계산 결과나 상호작용 결과는 Result Section에 생성되어 블록이 많을 경우 헷갈리지 않게 설계하였다.

II - 3 평가

사용 예시 1. 행렬 연산

Block Section	Result Section
$[3, 1, 2] \times [8, 7, 1]$	3 3

- 행렬의 곱 연산

$d \ e \ t \ ([3 3])$	3 3
-------------------------	-----

- 행렬의 여인수 계산

$i \ n \ v \ ([[1, 2], [3, 4]])$	
$[[-2, 1], [1.5, -0.5]]$	

- 행렬의 역행렬 계산

사용 예시 2. 함수 연산

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

$$g(x) = x - 1$$

$$f(g(6)) =$$

f u n c t i o n

f u n c t i o n

2 2 5

사용 예시 3. 기본 연산

$$1 - 4 + 5 = = 3$$

$$s i n(p i)$$

$$cos(pi \div 2)$$

$$5 >= 9$$

f a l s e

1 . 2 2 4 6 4 6 7 9 9

6 . 1 2 3 2 3 3 9 9 5 7

f a l s e

유튜브 링크

클릭

III 논의

아쉬운 점은 다 만들어 놓고 보니 기능이 기본 예시에서 크게 벗어나지 못한 것 같아 조금 아쉽다. 아이디어를 생각할 시간을 가져보고 구현해보려 했지만 구조를 처음부터 다시 짜야할 것 같아 엄두가 안나 포기했다. 시험기간과 일정이 겹쳐 시간이 더 충분했으면 하는 바램이 있다.

사용하다보니 `disassemble` 기능이 생각보다 불편하다는 생각을 했다. 그래서 중간이나 원하는 위치부터 자르는 기능을 추가해보려고 했지만 UI를 어떻게 설계해야 할지, 자르는 위치를 지정하는 부분에서 막혀서 해내지 못했다. 그래서 아직도 `disassemble` 부분에 대한 불편함이 해소되지 않은 듯 하다.

전반적인 자체 평가를 해보자면 기본 조작 계산기의 기능은 구현해서 성공했으나 딱히 눈에 띄게 발전한 차별점은 두드러지지 못해 사용성 면에서 더욱 개선이 필요하다고 할 수 있다. 이는 시간을 두고 발전시켜 나가야 할 사항들이다.