Tower of Hanoi

라인컴퓨터아트학원

C16ST15

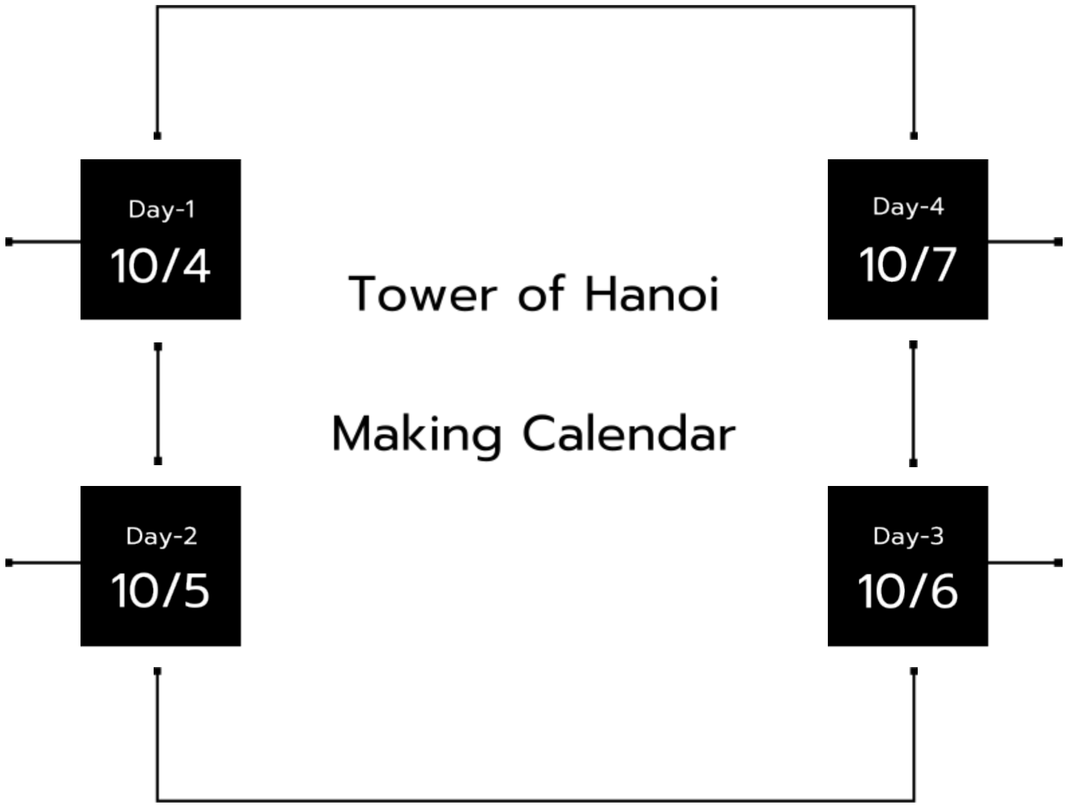
손 성 균

CONTENTS

CHAPTER 01

일정표

일정표



Day-1
10/4

프로젝트 시작

하노이 타워 기본 로직 파악
(하노이 타워 규칙성 및 이동 방향)

Day-2
10/5

스케치 및기본 디자인

스케치 작업 완료
기본 디자인 html과 css등 이용하여 완료

Day-3
10/6

로직 구현(자바스크립트)

하노이 타워 로직 구현하여
원반이 옮겨가는 회수, 방향에 대한
최소값 로직 구현

Day-4
10/7

최종 완료

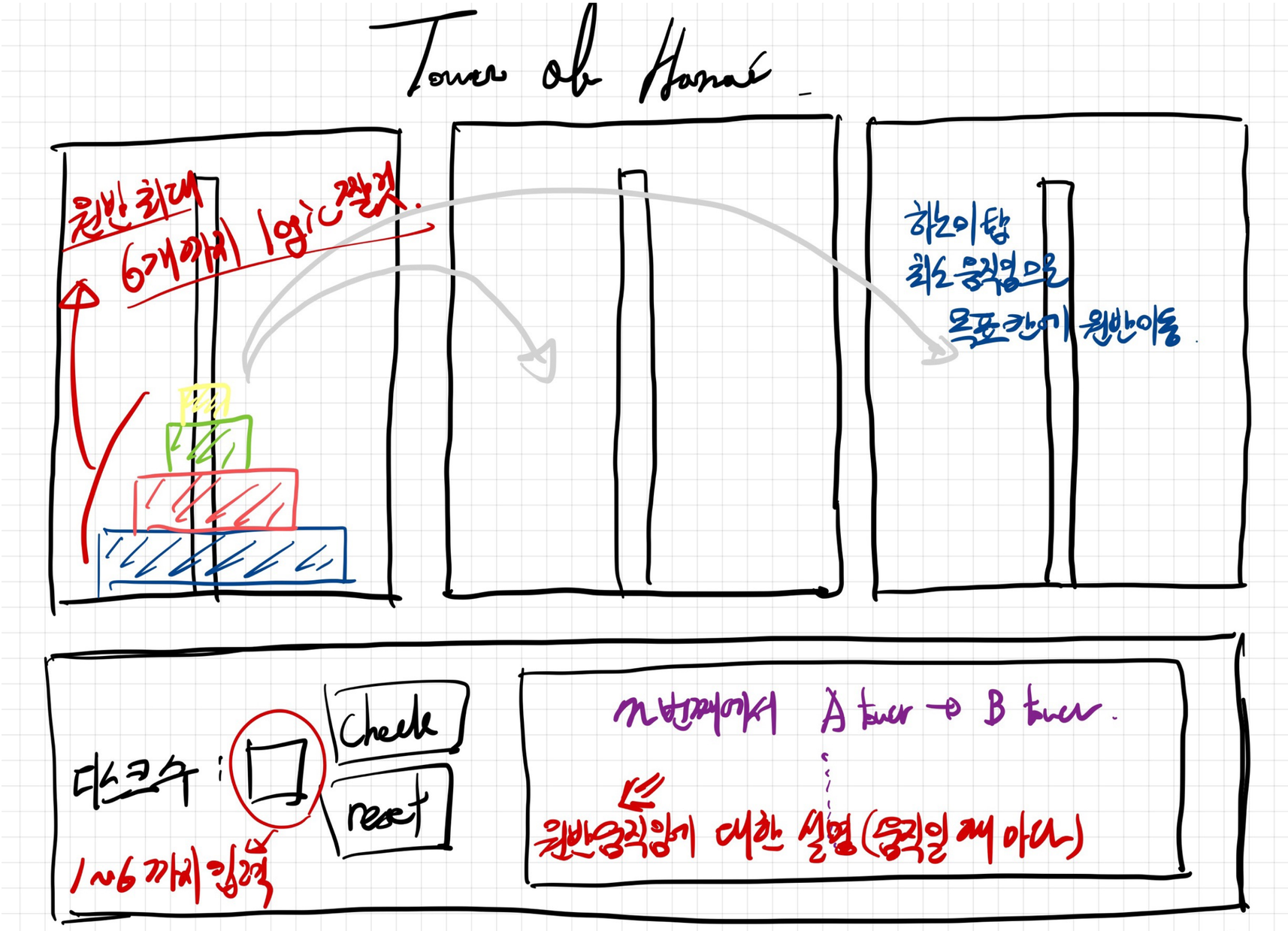
구현된 로직을 적용시켜 일정시간 단위로
원반이 옮겨가는 화면 구현 완료

CHAPTER 02

Sketch

Sketch

- 하노이의 타워는 총 세개의 공간(타워)이 존재함을 고려함.
- 하노이 타워는 원반들이 두번째, 세번째 타워를 통해 이동하고 세번째 타워에 첫번째 타워와 같은 모습으로 위치하여야 함을 고려함.
- 최대 6개의 원반까지 입력 받음.
- 원반 움직임에 대한 설명란을 추가하여 원반이 움직이는 횟수와 이동하는 타워를 안내.



CHAPTER 03

Design Layout

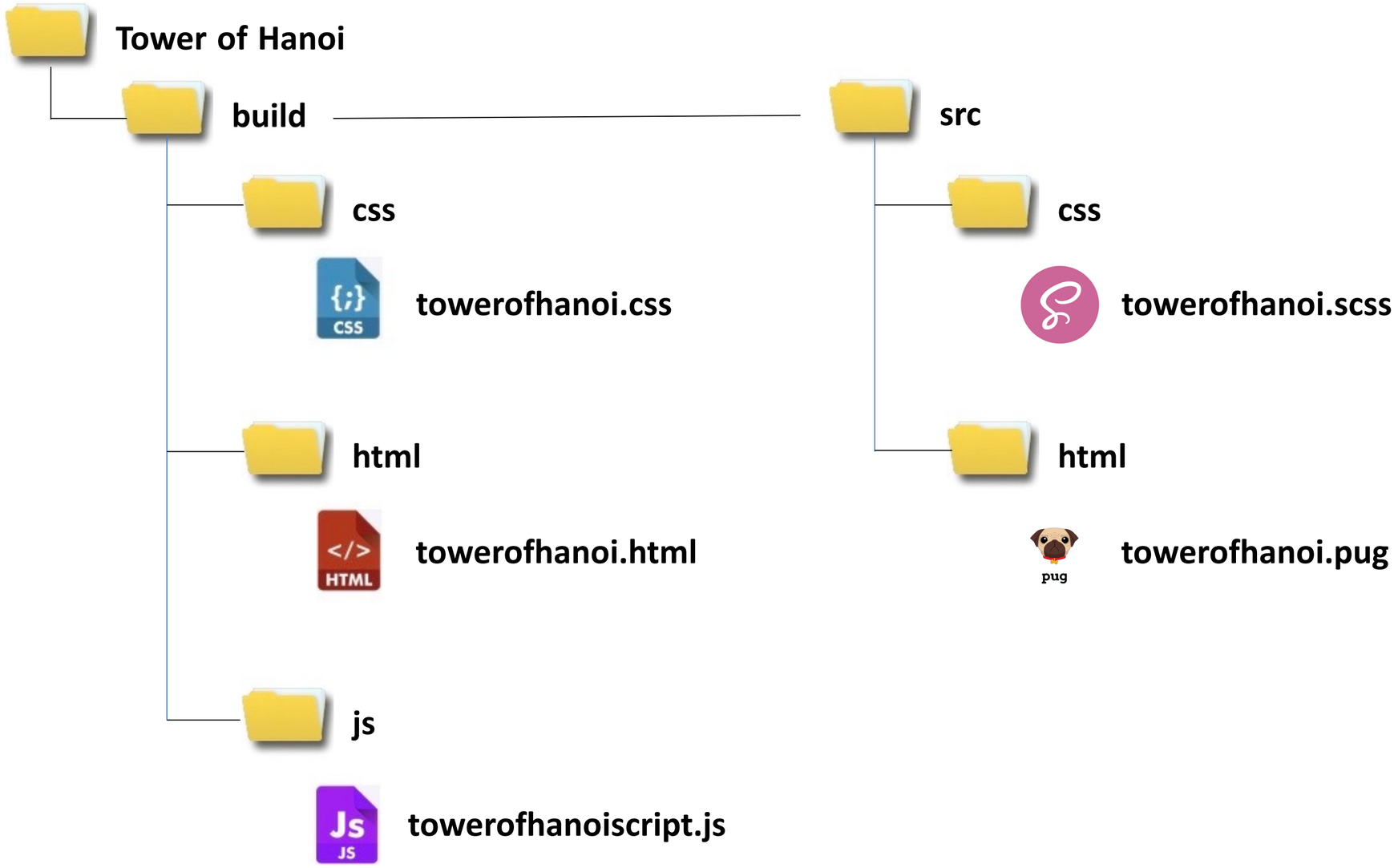
Design Layout

- 배경 이미지를 어두운 이미지로 설정하여 원반의 이동이 명확히 보이도록 함.
- 원반과 이동횟수 안내 문구 외에는 모든 요소에 투명도를 부여하여 움직임에 대한 가독성을 높임.
- 원반의 개수에 따른 최소 움직임을 나타내고, 원반이 움직이는 순서를 나타내는 문구창을 추가하여 원반의 움직임을 통한 결과 제공 뿐 아니라 누적된 결과를 제공함.

CHAPTER 04

Folder Direction

Folder Direction



CHAPTER 05

Code

Pug

```
body
  main(class="total_area")
    main(class="background_check")
    main(class="top_area") Tower of Hanoi
      section(class="title_area")
    main(class="hanoi_tower" id="hanoi_tower")
      section(class="first_bar" id="first_bar")
        section(class="stick_bar")
      section(class="second_bar" id="second_bar")
        section(class="stick_bar")
      section(class="third_bar" id="third_bar")
        section(class="stick_bar")
    main(class="check_area")
      section(class="input_area")
        div(class="input_total")
          div(class="intro_sentence") Disk  :
          input(type="number" class="input_number" id="input_number" autocomplete="off")
        div(class="button_area")
          div(class="confirm_button" id="confirm_button") 확인
          div(class="reset_button" id="reset_button") Reset
      section(class="check_numbering")
        div(class="inner_box" id="inner_box")
```

- 하노이 타워 3개를 작성하고, 내부에 stick_bar를 두어 마치 원반과 같이 생긴 모습을 부여함.
- 원반 개수 입력 받는 상자 및 하노이 타워 로직이 실행될 확인 버튼과 초기화 버튼을 작성

Sass

```
.contain1{
    @include sizeCheck(4vw, 7vh);
    background-color: tomato;
}
.contain2{
    @include sizeCheck(8vw, 7vh);
    background-color: lightcyan;
}
.contain3{
    @include sizeCheck(12vw, 7vh);
    background-color: antiquewhite;
}
.contain4{
    @include sizeCheck(16.5vw, 7vh);
    background-color: rgba(238, 225, 147);
}
.contain5{
    @include sizeCheck(20.5vw, 7vh);
    background-color: aquamarine;
}
.contain6{
    @include sizeCheck(25vw, 7vh);
    background-color: beige;
}
}
```

- 하노이 타워 내부에 생성될 원반을 미리 정의하여, 자바스크립트를 통해 생성될 때 해당 원반의 종류에 따라 크기 및 색상을 달리하는 코드 작성.

JS / Jquery

```
//하노이의 탑에 들어갈 숫자를 입력받기.
function input_hanoi_disk(start, input) {
  for(let i=0; i<input; i++) {
    start.push(input-i);
  };
};
//입력받는 숫자 ~6까지
function ban_number() { //인풋태그에 들어온 값을 받아오는 메소드
  $("#input_number").on("keyup", ()=>{
    //html 화면 표시가 가능한 개수를 1~6으로 설정했기 때문에
    //그외의 값이 들어오면 입력값 날리기
    if(!/^[1-6]{1}$/.test($("#input_number").val())){
      $("#input_number").val("");
    }
  })
};
```

- 하노이 탑에 존재하는 원반의 개수를 정하는 로직을 실행.
- 입력받는 값은 “1~6”까지로 한정하며 그 외의 값이 입력되면 입력 값이 사라지도록 코드 실행.

JS / Jquery

```
function start_hanoi(disk_count, start_bar, target_bar, support_bar) {
  counter++
  if(disk_count == 1){
      let move_disk = start_bar.pop();
      target_bar.push(move_disk);
      get_bar_array();
      get_disk_move(start_bar, target_bar);
  }
  if(disk_count >= 2){
      start_hanoi(disk_count-1, start_bar, support_bar, target_bar);
      let move_disk = start_bar.pop();
      target_bar.push(move_disk);
      get_bar_array();
      start_hanoi(disk_count-1, support_bar, target_bar, start_bar);
  }};
//하노이탑 내부 bar 출력하기(콘솔로 확인)
function get_bar_array() {
  console.log("1번 bar : " + start_count);
  console.log("2번 bar : " + support_count);
  console.log("3번 bar : " + target_count);
};
```

- 하노이 탑이 실행되는 메인 로직으로서 원반의 개수와 타워들을 각각 매개변수로 받아 재귀함수를 통한 하노이탑 최소이동회수로 완성하는 로직.

- 하노이탑 내부에 원반의 움직임을 확인할 수 있는 함수를 선언하여 로직의 정확성 파악

JS / Jquery

```
function get_disk_move(start_bar, target_bar) {
    flag_count++
    let start_num = 0;
    let target_num = 0;
    for(let i=0; i<total_arr.length; i++) {
        if(total_arr[i] == start_bar) {
            start_num = i+1;
        }
        if(total_arr[i] == target_bar) {
            target_num = i+1;
        }
    }
    console.log(flag_count+ "회에서 " + start_num + "번 bar에서 " + target_num + "번 bar로 이동");
    text_input.push([flag_count, start_num, target_num]);
};
```

- 하노이 탑 이동의 메인 로직
  내부에 원반 움직임 횟수 및 원반
  이동이 시작되는 타워와 도착하는
  타워에 대한 파악을 위한 함수
  실행

- 이후 해당 결과값을 통해
  변화하는 내용을 출력하기 위해
  새로운 배열에 담는 과정 추가

```
let output_text = () => {
    for(let i=0; i<text_input.length; i++) {
        setTimeout(() => {
            $("#inner_box").append(
                `<p class="append_text" id="append_text">${text_input[i][0]}번째 : ${text_input[i][1]}번 bar에서 ${text_input[i][2]}번 bar로 이동 </p>`
            );
            $("#inner_box").scrollTop($("#inner_box").prop("scrollHeight"));
        }, (i+1)*1000);
        if(i == text_input.length-1) {
            setTimeout("reset_data()", (i+1)*1100); //전체 다 돌고나면 resetData로 여태까지 했던 값들을 전부 초기화 처리하는 것.
        };
    };
};
```

JS / Jquery

- 원반 움직임 횟수 및 원반 이동이 시작되는 타워와 도착하는 타워에 대한 내용을 담은 배열을 통해(“text_input”) 해당 로직을 출력하는 함수 선언하여 실행.

```
let disk_make_A = () => {
  // console.log(memory_list[0][0]);
  for(let i=0; i<memory_list.length; i++){
    setTimeout(()=>{
      $(`#first_bar>div`).remove();
    },(i+1)*1000); //1초 간격으로 다음 배열로 넘어갈 때 기존에 있던 것을 1초 간격으로 없애야함.
    for(let j=memory_list[i][0].length-1; j>=0; j--) {
      setTimeout(() => {
        $("#first_bar").append(`<div class="disk_box contain${memory_list[i][0][j]}" id="contain${memory_list[i][0][j]}"></div>`)
      },(i+1)*1000);
    }
  }
}
```

JS / Jquery

- 하노이 탑 메인 로직 내부에 움직임을 횟수, 타워종류, 이동 될 때마다의 각 타워 내부 원반 갯수를 배열에 담아(memory_list) 해당 배열을 이용하여 원반 생성
- 원반 생성 시에는 1초 간격을 두고 생성과 삭제를 통해 마치 이동하는 듯한 효과 부여함.

CHAPTER 06

마무리

마무리

하노이 타워의 문제는 여러 보드게임에서 접해봤을 뿐 아니라, 전시회 등에서 설치 미술로도 접해본 기억이 있던 문제였습니다.

해당 문제는 늘 이리 옮기고 저리 옮겨도 답을 쉽게 찾을 수 없어 중간에 포기했던 문제였기에 처음 본 프로젝트를 수행할 때는 어떻게 해결해야할 지 고민이 많았습니다.

그러나 본 프로젝트를 진행하면서 하노이의 타워 역시 한 가지 로직에 의해 움직이며, 그 풀이는 정확한 순서와 방향만 가지면 원반이 몇 개가 있더라도 해결 할 수 있다는 것을 알게 되었습니다.

해당 규칙성을 이해하고 코드를 통해서 구현함에 있어 재귀함수라는 방법을 사용했고, 배열을 통해 결과값을 담아 화면 상에 구현할 수 있었습니다.

오랫동안 저를 괴롭혀 왔던 문제였지만 코딩을 통해 해결했다는 점에서 뿌듯한 감정을 느낄 수 있었고, 나아가 자바스크립트의 배열개념과 함수개념에 대한 이해를 다시한번 확립할 수 있었습니다.

사이트 연결 : http://kkms4001.iptime.org/~c16st15/portfolio/TowerOfHanoi%20project/build/html/towerofhanoi.html

THANK

YOU EVERYONE