프로젝트 보고서

전공: 학년: 학번: 이름:

1. 프로젝트 주제

숲(나무와 꽃)을 그리는 프로그램.

2. 프로젝트 목표

OpenFramework 를 이용해 숲을 그리는 프로그램을 만들고자 하였다. 마우스로 나무를 그릴 위치와 몸통의 길이를 선택하면 자동으로 나무를 그리도록 하고자 했다. 또한 그린 나무에 랜덤한 색의 꽃을 그리고자 하였다. 구현한 기능은 다음으로 요약된다.

- 선택한 위치에 나무 몸통의 길이를 정하고 나무를 그린다.
- 랜덤하게 나무를 그린다.
- 선택한 나무가 꽃을 피우게 하거나 꽃을 없앤다.
- 모든 나무가 꽃을 피우게 한다.
- 모든 나무의 꽃을 없앤다.
- 선택한 나무를 지운다.
- 모든 나무를 지운다.

실험 환경: visual studio 2019

3. 자료구조

3-1. Branch 구조체

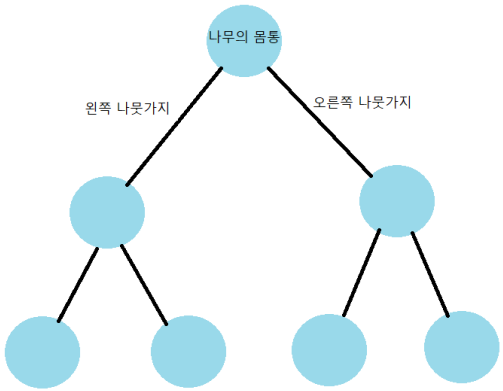

그림 1 [나무 이진 트리]

하나의 나무에 대한 정보는 Branch 구조체를 노드로 갖는 이진 트리로 저장된다. 각 노드는 나뭇가지에 대한 정보를 저장한다. 노드의 왼쪽 및 오른쪽 자식 노드는 부모 노드에 저장된 나뭇가지에서 90 도 방향으로 자란 왼쪽 및 오른쪽 나뭇가지를 상징한다.

struct Branch	
pair<double, double> bottom, top;	bottom: 부모 나뭇가지에서 자라난 위치를

	저장한다. top: 나뭇가지 끝이 위치하는 좌표를 저장한다.
double length;	나뭇가지의 길이를 저장한다.
int direction;	나뭇가지가 자란 방향을 저장한다. bottom에서 top 방향으로 자란다고 본다. 값은 1~4이며 다음 의미를 갖는다. 1: 위쪽 방향 2: 아래쪽 방향 3: 왼쪽 방향 4. 오른쪽 방향
int end;	값이 1이면 리프 노드임을 뜻한다.
Branch* left; Branch* right;	현재 나뭇가지에서 발아하는 왼쪽 및 오른쪽 나뭇가지 노드를 가리키는 포인터이다.

3-2. Node 구조체

나무에 대한 정보를 저장하기 위해 헤드 노드를 가지고 원형으로 연결된 연결 리스트를 저장한다. 임의의 노드 삽입과 삭제를 쉽게 하기 위해 연결 리스트에 정보를 저장하도록 선택했다. 연결 리스트의 각 노드는 Node 구조체로 구현된다.

struct Node	
Branch* tree;	나무의 정보를 저장한다. 나무 정보를 저장하는 이진 트리의 루트 노드를 가리키는 포인터이다.
int bloom_check[4];	나무의 꽃 정보를 저장한다. 0 번째 원소: 값이 1 이면 꽃이 피었음을 뜻한다. 1~3 번째 원소: 꽃의 색 정보를 저장한다.
Node* left; Node* right;	왼쪽 및 오른쪽 노드를 가리키는 포인터이다.

3-3. Mode 열거형

나무의 몸통을 자라게 하고 있는지의 여부를 저장하는 mode 변수의 값에 다음과 같이 이름을 부여했다.

enum MODE

MODE_TRUNK_START	몸통을 자라게 하고 있음을 뜻하는 값(0)이다.
MODE_TRUNK_END	몸통을 자라게 하고 있지 않음을 뜻하는 값(1)이다.

4. 변수 설명

Node* head;	나무를 저장하는 연결리스트의 헤드 노드를 가리키는 포인터.
int tree_count;	화면에 그려진 나무의 개수를 저장하는 변수.
int screenHeight	스크린 높이를 저장하는 변수.
int screenWidth	스크린 너비를 저장하는 변수.
int mode;	나무 몸통을 자라게 하고 있는지, 자라게 하는 것을 멈췄는지 저장하는 변수.
int trunk_length;	나무 몸통을 자라게 하고 있을 때 그 길이를 저장하는 변수. 프레임마다 값이 10 씩 증가한다.
int mouse_x;	나무의 몸통을 자라게 할 위치의 x 좌표를 저장하는 변수
Node* current_node;	현재 선택된 나무 정보를 저장하는 노드를 가리키는 포인터.
int watch;	감상 모드의 여부를 저장하는 변수. 0: 선택된 나무가 갈색으로 표시된다. 1: 아무 나무도 선택되지 않는다. (감상 모드)

FLOWERRAD 는 꽃을 표현하는 원의 반지름 값으로 5 로 설정되어 있다.

5. 함수 소개

double RandomDouble(double a, double b);	a 에서 b 사이 임의의 정수를 반환한다.
void GrowTree(double x, double len);	나무의 몸통 위치의 x 좌표와 길이 len 을 인자로 받아 하나의 나무를 생성한다. 나무 몸통에 대한 노드(루트 노드)를

	생성하고 GrowBranch()를 호출한다. InsertNode()를 호출해서 생성한 나무 정보를 연결 리스트에 넣는다.
`void GrowBranch(Branch** b);`	b 가지에 대해 왼쪽과 오른쪽 가지를 랜덤으로 뻗는다. 재귀적으로 호출하여 하나의 나무를 생성한다.
`void DeleteBranch(Branch** b);`	나뭇가지 노드 b 에 할당된 메모리를 해제한다. 재귀적으로 호출되어 결국 하나의 나무를 지운다.
`void DrawBranch(Branch* b);`	나뭇가지 b 를 화면에 그린다. 재귀적으로 호출되어 결국 하나의 나무를 그린다.
`void DrawFlower(Branch* b);`	나뭇가지 b 끝에 핀 꽃을 화면에 그린다. 재귀적으로 호출되어 결국 하나의 나무에 핀 꽃 모두를 그린다.
`void InsertNode(Branch** b, int bc[4]);`	b 를 루트 노드로 갖는 나무 이진 트리와 꽃 정보 배열 bc 를 저장하는 노드를 생성하고 연결리스트에 넣는다. 이때, 나무 몸통의 x 좌표 그리고 길이에 따라 오름차순 정렬을 유지한다.
`void DeleteNode(Node** cur);`	cur 노드를 연결리스트에서 삭제한다.
`void EraseList();`	연결리스트에 할당된 메모리를 해제하고 연결리스트 전체를 삭제한다. DeleteBranch()를 호출해서 각 노드에 저장된 나무 정보를 삭제한다.

6. 알고리즘

해당 프로그램은 1 초에 15 번씩 자동으로 호출되는 draw 함수와 입력받은 키에 반응하여 작업을 수행하는 여러 함수로 구성되어 있다.

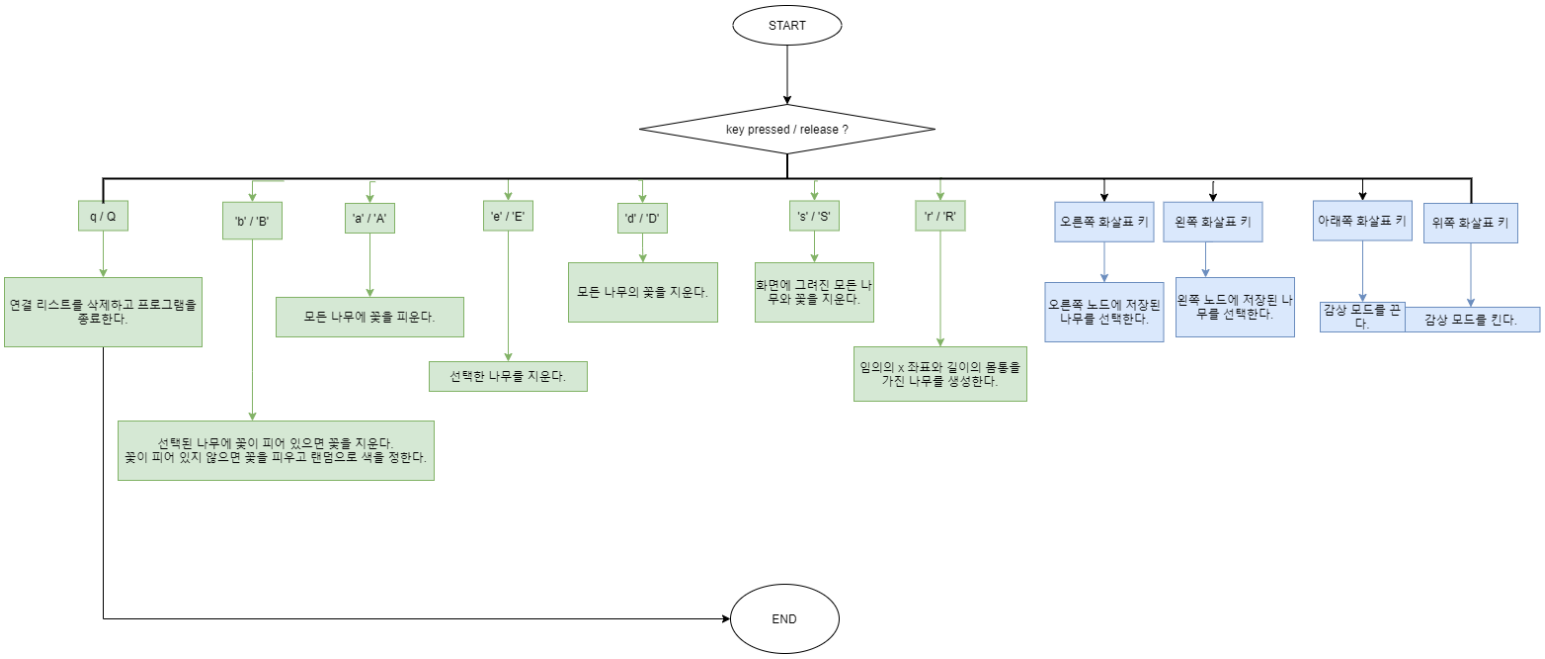

그림 2 [프로그램 구조도]

하나의 나무는 나무 몸통 노드를 루트 노드로 하는 이진 트리로 저장된다. 나무를 생성하거나 그릴 때, 이진 트리를 DFS 방법으로 탐색하면서 각 나뭇가지에 대해 작업을 한다. 따라서 하나의 나무를 생성하거나 그리는 시간 복잡도는 O(나뭇가지 개수)이다.

노드를 삽입할 때는 연결 리스트가 나무 몸통의 x 좌표와 길이를 기준으로 오름차순 정렬을 유지하는 위치를 찾아야 하므로 O(나무 개수)의 시간 복잡도를 가진다. 노드 삭제는 O(1)의 시간 복잡도를 지닌다.

나무 몸통의 최대 길이는 768 (스크린 높이) 이므로 이진 트리의 최대 높이는 10 이다. 따라서 이진 트리의 공간 복잡도는 O(노드 개수)= O(1023) 이하가 된다.

* 화면의 좌표는 다음과 같이 표현된다.

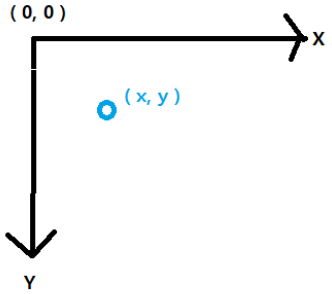

그림 3 [화면 좌표]

6-1. 나무와 꽃을 화면에 그리는 알고리즘

사용하는 함수

```
void draw()
/*
기능: 나무 몸통의 위치와 길이를 정하고 나무를 생성한다. 화면에 나무를 그린다.
*/
```

```
void DrawBranch(Branch* b)
/*
        입력:  Branch* b: 현재 나뭇가지
        기능: 나뭇가지를 화면에 그린다.
*/
```

```
void DrawFlower(Branch* b)
        /*
                입력:  Branch* b: 나뭇가지
                기능: 나뭇가지에 꽃을 그린다.
        */
```

그림 4 [나무 몸통을 자라나게 하는 화면]

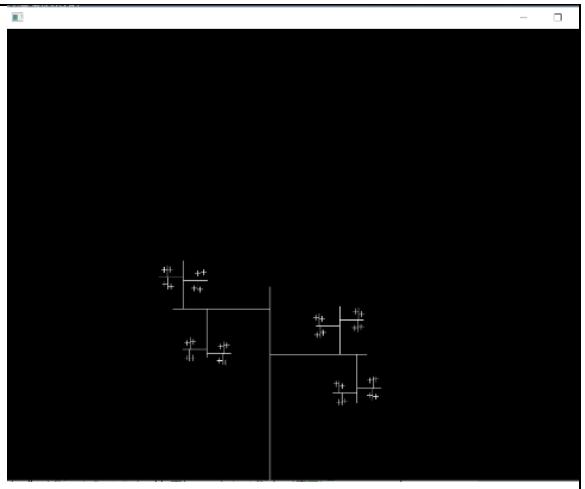

그림 5 [나무를 생성한 화면]

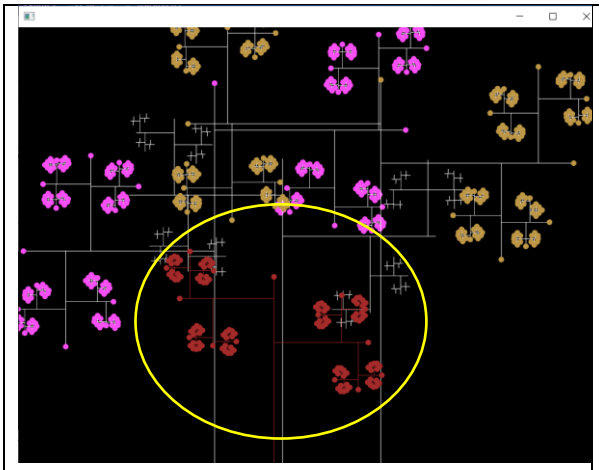

그림 6 [꽃을 핀 나무를 선택한 화면]

노란색 원 안에 선택된 나무가 있다. (노란색 원은 프로그램 화면에 포함되지 않는다)

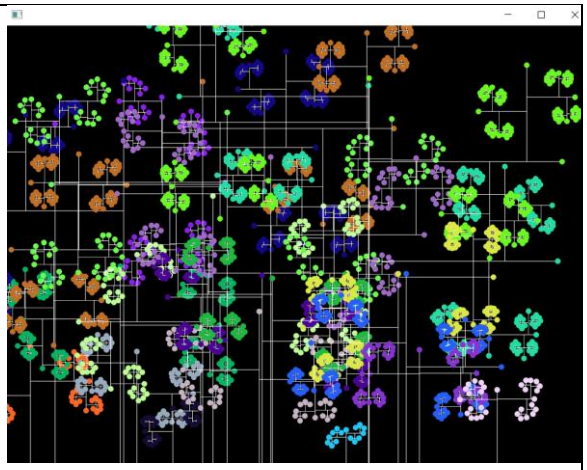

그림 7 [감상 모드 화면]

<draw()>

draw 함수에서 마우스 왼쪽 버튼을 이용하여 나무 몸통의 위치와 길이를 정하고 나무를 생성한다.

MODE_TRUNK_START 모드일 때:

draw 함수가 호출될 때마다 나무 몸통의 길이를 10 씩 증가시킨다. 갱신된 길이에 따라 화면에 나무 몸통을 그린다.

왼쪽 마우스 버튼이 눌리면, mouse_x 변수에 저장된 x 좌표 값과 trunk_length 에 저장된 길이 값에 따라 GrowTree()를 호출해서 나무를 생성한다. 나무 몸통을 자라게 하지 않는 모드로 바꾼다.

MODE_TRUNK_END 모드일 때:

왼쪽 마우스 버튼이 눌리면, 마우스의 x 위치 좌표를 저장하고 나무 몸통 길이를 0 으로 초기화한다. 나무 몸통을 자라게 하는 모드로 바꾼다.

모드에 상관없이 연결 리스트를 탐색하면서 각 나무에 대해 DrawBranch()를 호출하여 화면에 그린다. 꽃이 피었다고 저장되어 있으면 DrawFlower()를 호출하여 화면에 그린다.

선택되지 않은 나무들은 하얀색으로 그려진다. 감상 모드일 때 선택된 나무(와 꽃)을 갈색으로 화면에 표시한다.

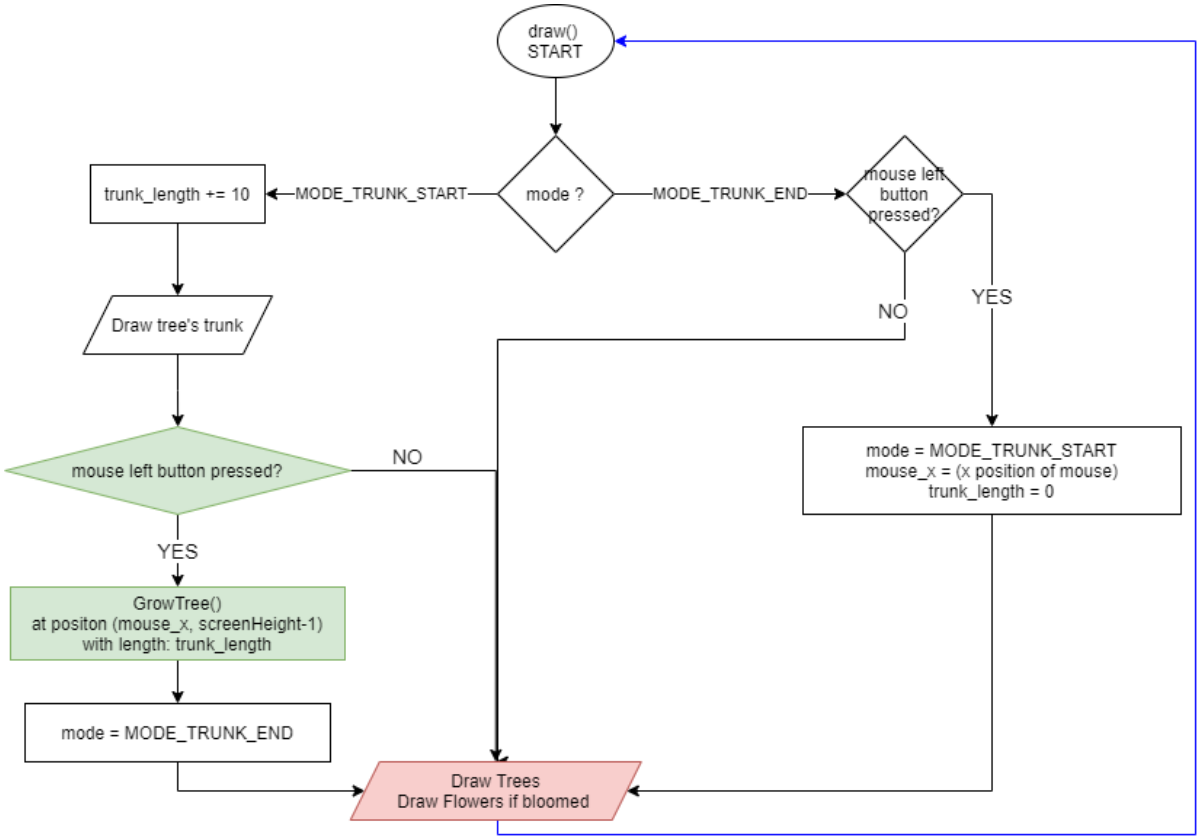

그림 8 [draw 함수 순서도]

<DrawBranch()>

나뭇가지의 bottom 좌표와 top 좌표를 양 끝으로 하는 선분을 화면에 그린다. 자식노드가 있으면 각 자식 나뭇가지에 대해 재귀적으로 호출한다.

<DrawFlower()>

나뭇가지의 위 끝인 top 좌표를 중심으로 하고 반지름이 FLWOERRAD 인 원을 화면에 그린다. 자식노드가 있으면 각 자식 나뭇가지에 대해 재귀적으로 호출한다.

6-2. 나무를 생성하는 알고리즘

사용하는 함수

```
void GrowTree(double x, double len)
/*
* 입력:
*      double x: 나무 몸통의 x 좌표
*      double len: 나무 몸통의 길이
* 기능: 나무의 나뭇가지 정보를 저장하는 이진 트리를 생성한다.
*/
```

```
void GrowBranch(Branch** b)
/*
       입력: Branch** b: 현재 나뭇가지
       기능: 현재 나뭇가지에 대해 왼쪽, 오른쪽 자식 나뭇가지(노드)를
생성한다.
*/
```

```
void InsertNode(Branch** b, int bc[4])
/*
* 입력:Branch** b: 현재 나뭇가지
*             int bc[4]: 현재 나뭇가지의 꽃 정보
* 기능: 현재 나뭇가지와 꽃 정보를 저장하는 노드를 생성하고 연결 리스트에
넣는다.
*/
```

<GrowTree()>

GrowTree 함수를 호출하여 나무의 몸통을 저장하는 노드를 생성한다.

나무 몸통의 아래 좌표: (x, 스크린 높이)

위 좌표: (x, 스크린 높이 – 몸통 길이)

나무 몸통 방향: 위쪽 방향(1)

리프 노드인지 여부(end): 아니다(0)

꽃 여부: {0,0,0,0}으로 초기화

GrowBranch 함수를 호출해서 나뭇가지 정보를 저장하는 이진트리를 생성한다.

InsertNode 함수를 호출해서 연결리스트에 나무 정보를 넣는다.

<GrowBranch()>

GrowBranch 함수는 재귀적으로 호출된다. 현재 나뭇가지 길이의 반이 꽃의 반지름(FLWOERRAD) 이하가 되면 자식 나뭇가지를 생성하지 않고 함수를 종료한다.

현재 나뭇가지의 방향에 양쪽으로 90 도 되는 방향으로 왼쪽과 오른쪽 자식 나뭇가지를 생성한다.

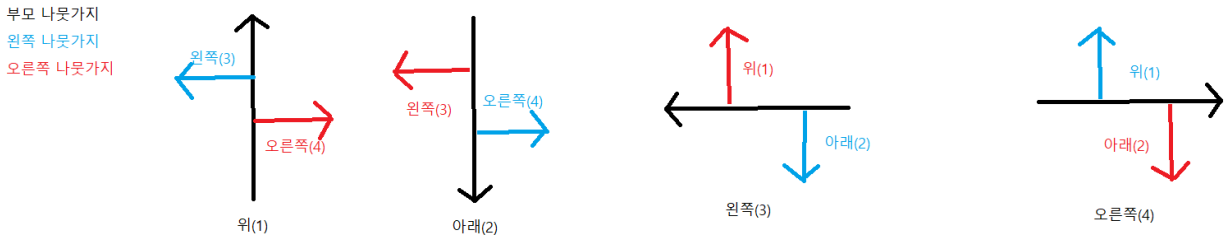

그림 9 [나뭇가지 방향]

자식 나뭇가지의 길이는 현재(부모) 나뭇가지 길이의 반이다. 자식 나뭇가지는 각자 부모 나뭇가지의 위 절반 부분에서 랜덤으로 정해진 위치에서 자라도록 한다.

<InsertNode>

연결 리스트가 노드에 저장된 나무 몸통의 x 좌표와 길이를 기준으로 오름차순 정렬을 유지할 수 있게 하는 위치를 찾고, 그 곳에 노드를 삽입한다.

6-3. 나무를 삭제하는 알고리즘

사용 함수

`void DeleteNode(Node** cur)`
`/*`
`* 입력:Node** cur: 삭제할 노드`
`* 기능: 노드를 연결 리스트에서 삭제한다.`
`*/`

`void ofApp::DeleteBranch(Branch** b)`
`/*`
`입력: Branch** b: 삭제할 나뭇가지`
`기능: 나뭇가지 노드의 메모리를 해제한다.`
`*/`

<DeleteNode()>

삭제할 노드를 연결 리스트에서 빼기 위해 앞 노드와 뒤 노드를 링크 포인터를 통해 연결한다. DeleteBranch 함수를 호출해서 삭제할 노드에 저장된 나무 이진 트리의

메모리를 해제한다. 삭제할 노드의 메모리를 해제한다. 이때, 삭제할 노드를 가리키던 currrent_node 는 앞 노드를 가리키도록 변경한다.

<DeleteBranch()>
자식 나뭇가지가 있으면 왼쪽 자식 나뭇가지와 오른쪽 자식 나뭇가지에 대해 재귀적으로 호출한다. 현재 나뭇가지 노드의 메모리를 해제한다.

7. 느낀 점 및 개선 사항

화면에 그릴 모형과 그 모형을 저장하기 위해 수치로 변환한 자료구조 사이에 괴리가 컸다. 나무를 각 나뭇가지의 양 끝 좌표와 길이를 저장하는 노드들로 이루어진 이진 트리로 변환하는 과정이 어려워서 각 나뭇가지마다 자식 나뭇가지가 2 개로 고정되어 있고, 나뭇가지 각도가 90 도인 단순한 나무 모양을 유지해야 해서 아쉬웠다.

현재는 그리는 나무는 부모 나뭇가지에 대해 90 도 방향으로 자식 나뭇가지가 자란다. 부모 나뭇가지에 대해 자식 나뭇가지들이 자라는 각도를 랜덤으로 정하고, 자식 나뭇가지 개수도 랜덤으로 정해서 더 풍성하고 다양한 형태의 나무를 그리도록 개선할 수 있다.

왼쪽 마우스 버튼을 사용해서 나무를 연속으로 생성할 때 너무 빠르게 클릭하면 나무 몸통이 제대로 자라지 않는 경우가 있다.