10주차 결과보고서

전공: 수학/컴퓨터공학 학년: 3학년 학번: 20181288 이름: 윤성호

<setup 함수에서 초기화>

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 1]

모든 과정이 진행되기 이전에 오픈프레임워크는 초기화 과정을 setup함수에서 진행해줄 수 있다. 초기화 과정은 아래와 같다. [사진 1]은 실습시간에 작성한 setup 함수이다.

* ofSetFrameRate(15) : 초당 프레임수를 15개로 제한한다.
* ofSetBackgroundAuto(False) : 매 프레임마다 그려지는 그림이 지워지지 않도록 한다.
* ofBackground(255,255,255) : 뒷 배경색을 하얀색으로 지정한다.
* ofSetLineWidth(4) : 그릴 선의 굵기를 4로 설정한다.
* Draw\_flag = 0 : 점과 선분을 그릴지의 여부를 담는 변수. 1이면 그리고, 0이면 그리지 않는다.
* Load\_flag = 0 : 입력 데이터를 받아왔는지의 여부를 담는 변수. 1이면 받아온 것이고, 0이면 받아오지 못한 것이다.
* Water\_flag = 0 : 물이 흐르는지의 여부를 담는 변수. 1이면 물이 흐르고 있는 것이고, 0이면 물이 흐르지 않는 것이다.
* Dot\_diameter : 10.0f : 물이 나올 점은 작은 원으로 표현한다. 이 원의 반지름을 나타내는 변수이다.

<update 함수>

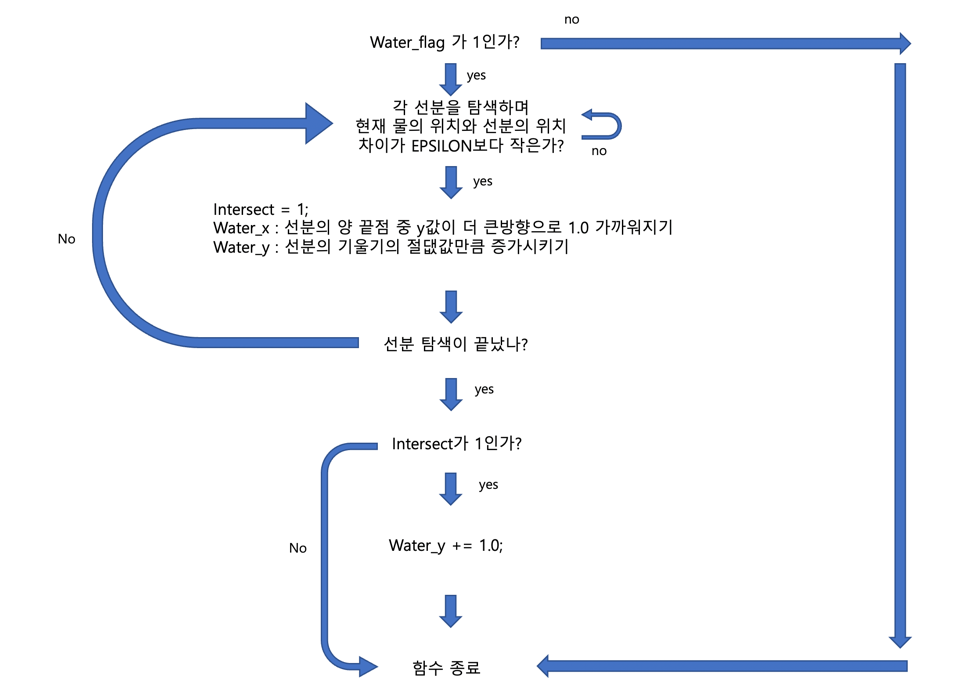
텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 2]



[사진 3]

오픈프레임워크에서 update 함수는 매 프레임마다 실행되는 함수이다. 그림을 그리는 부분이 아닌 부분에서 값을 변경해야 되는 상황에서는 update 함수안에 코드를 작성하면 된다. 실습시간에서 update 함수에 작성해야 하는 부분은 물이 흐르는 파트이다. 매 프레임마다 물의 위치를 변화시켜서 물이 흐르는 듯한 애니메이션을 구현하기 위해서는 update 함수 안에서 물의 위치 값을 변경해주어야한다. 먼저, 물이 흐르지 않는 상황에서는 물의 값을 변화시켜 줄 필요가 없으므로, 물이 흐르는 상황에서만 값의 변경이 일어나도록 한다. 즉, water\_flag가 1인 상황에서 값을 변경해준다. 물은 선분을 만나지 않는 상황에서는 수직으로 내려오다가 선분을 만나면 선분을 타고 내려오도록 한다. 이를 구현하기 위해 먼저, intersect 변수를 두고 0으로 초기화한다. 만약 선분과 만났다면 intersect를 1로 설정하도록 코드를 작성하였다. 선분과 만났다는 것을 판단하기 위해서는 모든 선분들을 살펴보며 직선의 방정식을 구성하고, 주어진 선분의 구간에서 물의 좌표를 대입하였을 때 0이 되는지의 여부로 만나는 것을 판단하였다. 그러나, c언어에서는 실수를 표현하는 방식으로 부동소수점 방식을 선택한다. 이는 모든 실수를 정확히 표현하는데는 오차가 생길 수 밖에 없다. 따라서, 물의 좌표를 직선의 방정식에 대입하였을 때 완전히 0이 되는 값이 없을 수도 있다. 따라서, 물의 좌표를 직선의 방정식에 대입한 값의 절댓값이 완전한 0은 아니더라도 어느정도 차이 이하의 값이라고 계산되면 만났다고 판단하였다. 절댓값은 fabs함수를 사용하였다. 여기서 어느정도 차이라고 이야기한 값을 EPSILON 이라고 정의하였으며 헤더파일에 값은 0.5로 설정하였다. 결국, 물이 선분과 만나지 않았다면, 물의 y좌표를 1.0 만큼 증가시켜주었고, 물이 선분과 만났다면 만난 선분의 양 끝 두 점의 y값 중 더 큰 점 쪽으로 물이 이동하도록 하였다. 이는 물의 x값은 y값이 더 낮은 쪽으로 1.0 가까워지도록 하였고, y값은 기울기의 절댓값만큼 증가하도록 하였다. [사진 2]는 실습시간에 작성한 update 함수이고, [사진 3]은 update 함수의 플로우차트이다.

<Draw 함수>

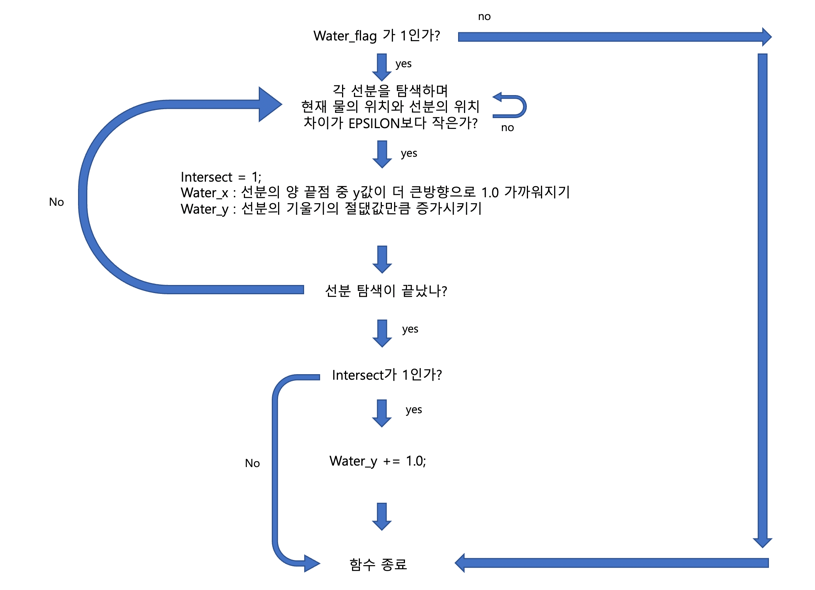
텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 4]



[사진 5]

오픈프레임워크에서 draw 함수는 update 함수와 마찬가지로 매 프레임마다 새롭게 실행되는 함수이다. Update 함수에서는 데이터 값 변경 등 그림 그리는 것과 관련이 없는 과정을 작성하였다면, draw함수에서는 실제 화면에 그리는 과정을 코드로 작성한다. 먼저, 물이 나올 점, 물이 타고 내려올 선분, 물이 흐르는 것을 그리기 이전에 화면 초기화과정부터 진행해준다. 선분을 적당한 갈색으로 그려주기 위해 ofSetColor(127, 23, 31)로 설정해주었다. (127, 23, 31)은 갈색을 RGB로 나타낸 것이다. 그리고 이 색으로 화면의 윗부분과 아랫부분에 직사각형을 그려주어, 내부 화면과 외부 화면을 구분해주었다. 그 이후 그릴 선의 굵기는 5로 설정하기 위해서 ofSetLineWidth를 5로 설정해주었다. Draw 함수는 총 2개의 조건문으로 구성되어 있는데, 하나는 draw\_flag가 1일 때 실행되는 부분과, water\_flag가 1이고 바닥에 닿지 않은 상황에서 실행되는 부분이 그것이다. Draw\_flag가 1이면 입력 파일에서 받아온 모든 선분들이 저장되어 있는 LINES 배열과 모든 점들이 저장되어 있는 DOTS 배열을 탐색하며 그려준다. 선분은 ofDrawLine함수를 이용해 그려주고, 점은 ofDrawCircle을 이용해 그린다. 점을 그릴 때는 dot\_flag에 해당하는 점은 빨간색으로 그려주고, dot\_flag에 해당하지 않는 점은 검은색으로 그려준다. Water\_flag가 1이고 물이 바닥에 닿지 않은 상황에서는 물을 파란색으로 그려준다. 파란색은 오픈프레임워크의 색상 객체를 이용해 myBlue라고 정의하였다. 그리고 ofSetColor로 그림 그릴 색을 파란색으로 설정해주고, 물을 반지름이 3인 원으로 그려준다. [사진 4]는 실습시간에 작성한 draw 함수이고, [사진 5]는 작성한 draw 함수의 플로우차트이다.

<’L’ key 눌러서 Input 받기>

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 6]

텍스트이(가) 표시된 사진

자동 생성된 설명

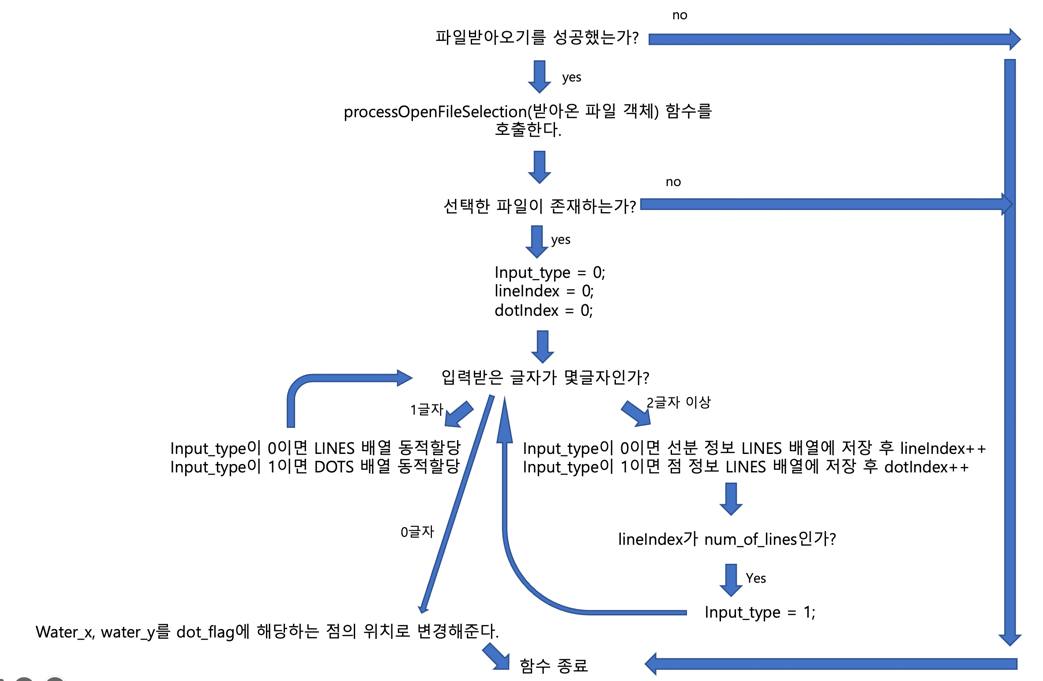
텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 7]



[사진 8]

키보드에서 “L” key를 입력받은것을 감지하도록 하기 위해 keyReleased 함수 안에 코드를 작성하였다. keyReleased 함수는 키보드 입력이 떼어지는 순간 실행되게 된다. “L” key를 누르면 먼저 선분의 개수를 초기화해준다. 그리고 ofFileDialogResult 객체를 정의하여 파일 열기를 시도한다. 만약, 파일을 열었다면 processOpenFileSelection함수를 이용해 파일 내부의 정보를 코드상의 LINES 배열과 DOTS 배열에 저장해주고 load\_flag를 1로 바꿔준다. Load\_flag는 점, 선분 정보를 불러오는 것을 성공하였다면 1로 설정해주고, 불러오는 것을 실패하였다면 0을 설정해준다. 이제 processOpenFileSelection 함수를 살펴보자. 파일 불러오기를 성공하였다면 점, 선분의 정보를 LINES 배열과 DOTS 배열에 저장한다. Input\_type 변수가 0이면 선분을 저장하고 있는 중이라는 의미이고 1이면 점을 저장하고 있는 중이라는 것을 의미한다. 그 상황에서 파일을 한 줄씩 읽어들이는데 만약 1글자를 읽었다면 이는 점 또는 선분의 개수를 의미하고, 2글자 이상을 읽었다면 이는 점 또는 선분의 정보를 읽고있다는 것을 의미한다. 먼저 1글자를 읽는 상황에서 input\_type이 0이고 num\_of\_lines 가 -1이라면 그 1글자는 선분의 개수를 의미한다. 선분 정보를 저장하는 이차원 배열인 LINES 배열을 동적할당해주고, 각각의 행의 첫번째 원소 또한 길이 4인 일차원 배열을 동적할당 해준다. 선분정보는 (x1, y1, x2, y2)로 주어지기 때문이다. lineIndex라는 변수도 설정하고 0으로 초기화해주어 LINES 배열에 몇개의 선분 정보가 들어가있는지 매순간 체크할 수 있도록 프로그램을 작성하였다. 그렇게 선분 정보를 받아오다가 num\_of\_lines가 linedIndex 와 같아진다면 모든 선분을 다 받아온 것이기 때문에 input\_type을 1로 변경해준다. 점을 받아올 차례이기 때문이다. 점들도 마찬가지로 1글자가 들어왔을 때는 점의 개수를 의미하므로 DOTS 라는 이차원 배열을 동적할당해주고, 각각의 행의 첫번째 원소 또한 길이 2인 일차원 배열을 동적할당 해준다. 점 정보는 (x1, y1)으로 주어지기 때문이다. dotIndex라는 변수도 설정하고 0으로 초기화해주어 DOTS 배열에 몇개의 점 정보가 들어가있는지 매순간 체크할 수 있도록 프로그램을 작성하였다. 그리고 마지막으로는 dot\_flag번째 인덱스에 들어가있는 점이 현재 선택된 점이라고 하고있기 때문에 물의 시작점을 현재 선택된 점이라고 볼 수 있다. 따라서 물의 x좌표와 y좌표를 dot\_flag의 x좌표와 y좌표로 변경해준다. [사진 6], [사진 7]은 실습시간에 작성한 ““L” key를 눌러 input받기” 과정을 코드로 작성한 것이고, [사진 8]은 ““L” key를 눌러 input 받기” 의 플로우차트이다.

<’D’ key 눌러서 draw 하기>

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 9]

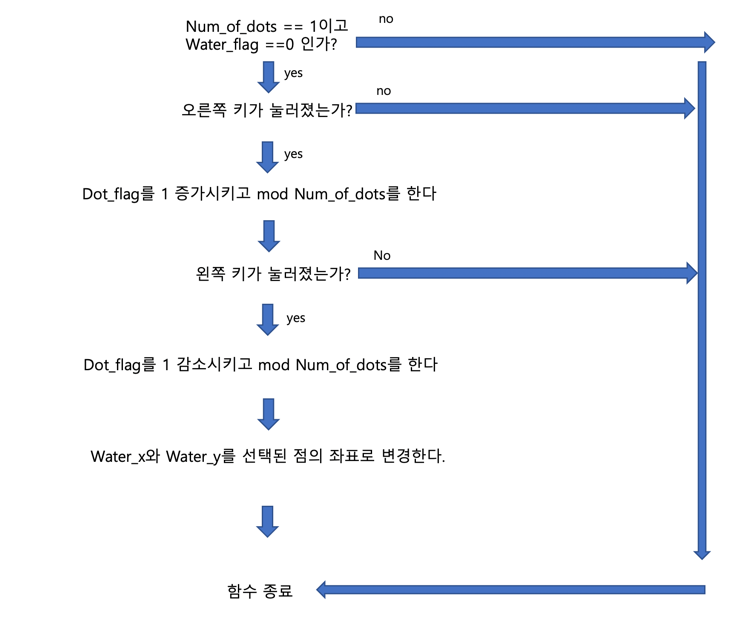
그림 그리기는 정보가 불러온 이후에 진행해야 한다. 따라서 load\_flag가 1이 아니라면 함수를 종료하고, 1일때 실행되도록 한다. 해야할 일은 draw\_flag를 1로 설정해주는 것으로 과정이 종료된다. 매 frame마다 update함수와 draw함수가 실행되는데 draw\_flag가 0으로 초기화되어있어서 점과 선분이 그려지지 않았었는데, d를 눌러서 draw\_flag를 1로 설정해주면 draw함수에서 조건문 내부로 들어갈 수 있게 되어 점과 선분을 그리게 된다. [사진 9]은 실습시간에 ‘D” key 눌러서 draw 하기 과정을 코드로 작성한 것이다. update함수와 draw 함수의 플로우차트는 각각 [사진 3], [사진 5]를 통해 확인할 수 있다.

<화살표로 물이 나올 지점 지정하기>

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 10]



[사진 11]

키보드의 화살표 버튼을 눌러서 물이 나올 지점을 지정해주는 과정이다. 키보드 입력이 필요하므로 keyReleased 함수에 코드를 작성하였다. 물이 나올 지점을 지정해주기 위해서는 일단 물이 나올지점이 있어야하고, 물이 흐르지 않는 상황이어야 한다. 이를 체크하기 위해 점의 개수를 저장하는 변수인 num\_of\_dot이 0이 아니면서, 물이 흐르는 지의 여부를 저장하는 플래그인 water\_flag가 0일때 물이 나올 지점을 변경할 수 있도록 한다. Water\_flag는 1이면 물이 흐르고 있는 상황, 0이면 물이 흐르지 않는 상황을 의미하도록 프로그램을 작성하였다. 이러한 조건문에 들어 온 이후에는 눌러진 키가 왼쪽 화살표인지 오른쪽 화살표인지 확인한다. 현재 선택된 점은 dot\_flag라는 변수에 저장되어있고, 이 값을 변화시켜서 물이 나올 지점을 변경한다. 왼쪽 화살표를 누르면 dot\_flag를 1 감소시키고, 오른쪽 화살표를 누르면 dot\_flag를 1 증가시킨다. 물론, 이 증감연산은 modular 연산이기때문에 증감연산을 진행한 후에는 num\_of\_dot으로 나누어준 나머지를 값으로 배정받는다. 이동이 되었다면 이동이 되었다는 메세지를 콘솔창에 출력해주고, ‘S’ key를 누르는 것을 대비하여, 현재 물의 x좌표와 y좌표를 각각 water\_x, water\_y라는 변수로 설정해주고, 이 변수에 점의 좌표를 대입해준다. 이렇게 준비가 되어있으면, ‘S’ key를 눌렀을 때 현재 점부터 물의 움직임이 시작 될 수 있다. [사진 10]은 실습시간에 “화살표로 물이 나올 지점 지정하기” 과정을 작성한 코드이고, [사진 11]은 “화살표로 물이 나올 지점 지정하기”의 플로우차트이다.

<’S’ key 눌러서 물 내려오게 하기>

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 12]

키보드의 “S” key를 눌러서 지정된 점을 시작으로 물을 내려오게 하는 과정이다. 먼저, 물의 모양은 작은 원이다. 오픈프레임워크는 그림이 그려진 흔적을 매 frame마다 지우기 때문에 물이 자취를 남기며 내려오게 하기 위해서는 ofSetBrackground(False)로 설정해주어야 한다. 그리고, draw\_flag가 1이라면, 즉 점과 선분이 이미 그려진 상태라면 water\_flag을 1로 설정해주고, draw\_flag를 0으로 설정해준다. Water\_flag을 1로 설정하는 이유는 물이 흐르기 시작하기 때문이고, draw\_flag를 0으로 설정해주는 이유는 점과 선은 매 frame마다 다시 그리게 하지 않기 위함이다. 이렇게 설정을 해놓으면 매 Frame마다 updateg함수와 draw함수가 알아서 실행이 되며 그림을 그리게 된다. [사진 12]는 ““S” key 눌러서 물 내려오게 하기” 과정 중 keyPressed함수에 작성 내용이다. update함수와 draw 함수의 플로우차트는 각각 [사진 3], [사진 5]를 통해 확인할 수 있다.

<’E’ key 눌러서 물 멈추게 하기>

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 13]

키보드의 “E” key를 눌러서 물이 내려오는 것을 멈추게 하는 과정이다. 키보드의 입력을 인식하기위해서 keyPressed함수 내부에 코드를 작성한다. 물이 내려오고 있을 때는 점과 선분은 이미 그려져 있는 것이고 내려오는 물이 계속 새롭게 그려지고 있다. 이때, 화면에 그렸던 흔적을 남기지 않기 위해 ofSetBackgroundAuto(true)로 설정해준다. 이렇게 설정해주면 화면에는 아무것도 남지 않게 된다. 물을 삭제해주기 위해서는 일단 물이 그려지고 있어야 하므로 water\_flag이 1이거나 점과 선분이 그려져 있어야 하므로 draw\_flag이 1이어야 한다. 그러한 상황이 아니면 물을 삭제할 수 없기 때문이다. 이러한 상황이면 물을 멈추게 해야하기 위해 water\_flag를 0으로 바꿔주고, draw\_flag는 1로 바꿔준다. 이렇게 되면 물은 삭제되고 점과 선분만 그려진다. [사진 13]은 실습시간에 ““E” key 눌러서 물 멈추게 하기” 과정을 작성한 코드이다. update함수와 draw 함수의 플로우차트는 각각 [사진 3], [사진 5]를 통해 확인할 수 있다.

<’Q’ key 눌러서 메모리 해제하기>

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 14]

키보드의 “q” key를 눌러서 동적할당 된 메모리를 해제하는 과정이다. 키보드 입력을 인식하기 위해 오픈프레임워크의 keyPressed함수에 코드를 작성하였다. Q를 입력하면 일단 draw\_flag을 0으로 설정한다. 그려진 그림을 지워주기 위함이다. 프로그램의 시작부분에서 입력을 받은 이후 선분 정보를 저장하는 LINES 배열과, 점 정보를 저장하는 DOTS 배열이 동적할당 되었다. 이 두 배열은 new 키워드를 이용해 동적할당했기 때문에 delete 키워드를 작성하여 메모리를 해제 해 주어야한다. 또한, 이차원 배열을 해제해주는 것이기 때문에 delete[] 키워드를 이용해야한다. 메모리가 해제되었다면 메모리 해제가 성공했다는 메세지를 콘솔창에 출력해주었다. [사진 14]는 “”Q” key를 눌러서 메모리를 해제하기” 과정을 작성한 코드이다. update함수와 draw 함수의 플로우차트는 각각 [사진 3], [사진 5]를 통해 확인할 수 있다.