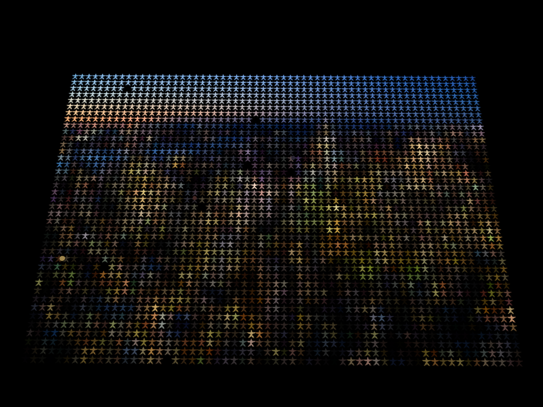
컴퓨터공학 설계 및 실험1 기말 프로젝트 보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20201635 이름: 전찬

**1. 프로젝트 목표 및 실험 환경**

다양한 사진에 대해 그 그림을 3d waterfall을 활용하며, 일정 크기(20x20) 형태로 분할해 원하는 unit이 반복되는 그림을 만들어낸다. 아래는 예시와 같다.

<실행 전 사진> <실행 이후 사진>

또한 실험 환경은 xcode에서 진행했으며, 실행 장비는 아래와 같다.



**2. 각 변수에 대한 설명**

1. ofImage image, main, unit, end

위 변수들은 ofImage를 활용해 사진을 load하며, 용도에 맞게 사용하는 변수이다. image변수는 원하는 사진의 가로와 세로 크기를 얻기 위해 사용되며, 나머지 변수들은 프로그램 화면을 보여주기 위해 사용된다.

2. int height, width

위 변수들은 ofImage image 변수에서 얻어낸 가로와 세로를 저장하는 변수이다.

3. OfEasyCam view

위 변수는 OfEasyCam을 활용해, 3차원으로 물방울이 떨어지는 형태와 그림이 그려지는 형태를 보게 해주는 변수이다.

4. bool\*\* start, int\*\* water\_loc, int\*\*\* color\_array

각각의 변수는 20x20으로 분할한 (width/20 \* height/20)의 각각에 영역에 대한 정보를 표시하는 변수들이다. 각각 변수들은 동적할당을 받으며, start는 그 영역에서 물이 떨어지기 시작했는지, water\_loc은 현재 물의 위치가 어디인지, color\_array는 각각의 영역에 대해 r g b 색의 평균값이 몇인지를 저장하고 있는 변수들이다.

5. ofPixels pixel\_data

위 변수는 color\_array에 r g b 값을 각각 저장하기 위해 필요한 변수이다.

6. int unit\_flag, execute\_flag, end\_flag

위 세 변수는 프로그램을 control 하기 위해 사용되는 변수이다. unit\_flag = 1이면 20x20 unit의 형태를 변경시킬 수 있으며, execute\_flag = 1이면 waterfall drawing이 실행되는 중이며, end\_flag = 1이면 동적할당 해제와 함께 프로그램의 종료 화면을 보여준다.

7. int select\_block[20][20]

현재 20x20의 unit 중 선택된 부분을 저장하는 변수이다.

8. const char\* filename

waterfall drawing을 실행시킬 사진의 이름이다. 기본은 “Gogh.png”로 되어 있으며, 사용자가 원하는 사진의 이름을 입력시킬 수 있다. (먼저 data 디렉토리 안에 원하는 사진을 집어넣어야 한다.)

**3. 자료구조 및 알고리즘 설명**

이번 프로젝트에서 가장 중요한 알고리즘은, width/20 \* height/20의 각 위치에서 물방울이 랜덤으로 떨어지며, 물방울이 다 떨어지면 (position = 0) 그 부분에 해당되는 사진에 unit의 모양에 따라서 색을 칠해 주어야 하는 것이다. 이를 구현하기 위해서 사용된 자료구조는 2, 3차원 동적 할당된 배열이다. 배열로 구현한 이유는, 첫 번째로 동적 할당으로 임의의 image size마다 width와 height를 얻어서 동적 할당함을 통해, 다양한 size를 가진 사진들에 대해서 프로그램이 공통적으로 적용될 수 있도록 하기 위해서 동적 할당을 사용했다. 두 번째로는 이번 실습에서 필요한 data가 sparse 형태가 아닌, dense matrix(배열의 모든 원소를 다 사용한다.) 이기 때문에 2차원 배열을 통해서 구현해주면 효율적이기 때문에 위 자료구조를 사용했다.

위에서 설명한 것과 같이, 이번 프로젝트에서 가장 중요한 알고리즘은 각 위치에서 물방울이 랜덤으로 떨어지며, 다 떨어진 경우 사진에 색을 칠해 주어야 하는 것이다. 이를 구현하기 위해서, openframeworks 의 update() 부분과 draw() 부분을 주로 활용했다. 알고리즘은 다음과 같다.

1. update를 통해서 물방울이 아직 떨어지지 않은 위치는 일정 확률로 물방울을 시작한다.

2. update를 통해 물방울이 시작된 경우 계속적으로 각 배열의 원소에 위치에 해당되는 물방울을 높이가 0이 될 때까지 떨어트려준다.

3. draw를 통해서 그 위치에 물방울이 시작되었으며, 물방울의 높이가 0인 경우 그 위치에 적절한 색을 unit 형태로 그려준다.

4. 위 방식을 계속해서 반복한다.

위 알고리즘을 통해서 물방울이 떨어지며, 그림이 그려지는 형태를 표현할 수 있다. 추가적으로, 위 알고리즘의 시간 복잡도는 O(width\*height) 형태이다. 이는 각 배열의 위치의 물방울마다 한 번씩 체크해 주기 때문이다.

**4. 각 함수에 대한 설명**

이번 프로젝트에서는 총 4가지의 함수를 사용했다. 각 함수들은 아래와 같다.

1. void arraymalloc()

위 함수는 위에서 설명한 세 변수, start, water\_loc, color\_array 에 대해 동적 할당을 받는 함수이다. 추가적으로 각 배열의 원소들을 모두 0으로 초기화한다.

2. void getavg()

위 함수는 color\_array[width/20][height/20][3]에 그림의 색의 평균값을 저장해주는 함수이다. 평균값을 구할 때는 pixel\_data 변수를 통해서, width/20 \* height/20 의 각 20\*20 정사각형마다, 모서리 4점의 평균을 color\_array[][][3]에 r ,g, b 값을 차례로 저장해준다.

3. void drawfigure()

execute\_flag = 1인 경우 실행되는 함수이다. 물방울의 시작과 위치를 판단하여, 시작 = 1, 위치 = 0 인 경우 각 위치에 unit과 적절한 색(getavg()에서 얻어낸 평균 색)을 토대로 그림을 그려준다.

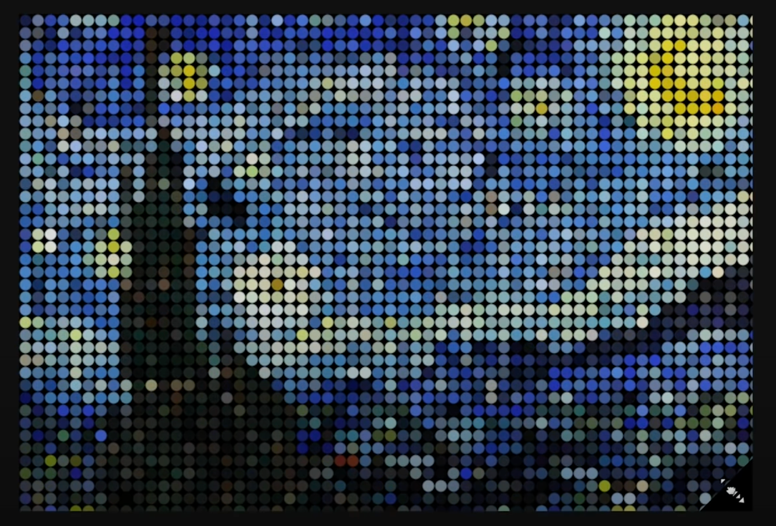
4. void waterfall()

물방울의 떨어짐을 구현하는 함수이다. 일정 확률로 물방울을 시작 시키며, 위치를 0이 될 때까지 아래로 내려주는 함수이다.

**5. 본인이 어떻게 창의적 구현을 하였는지에 대해 구체적으로 설명**

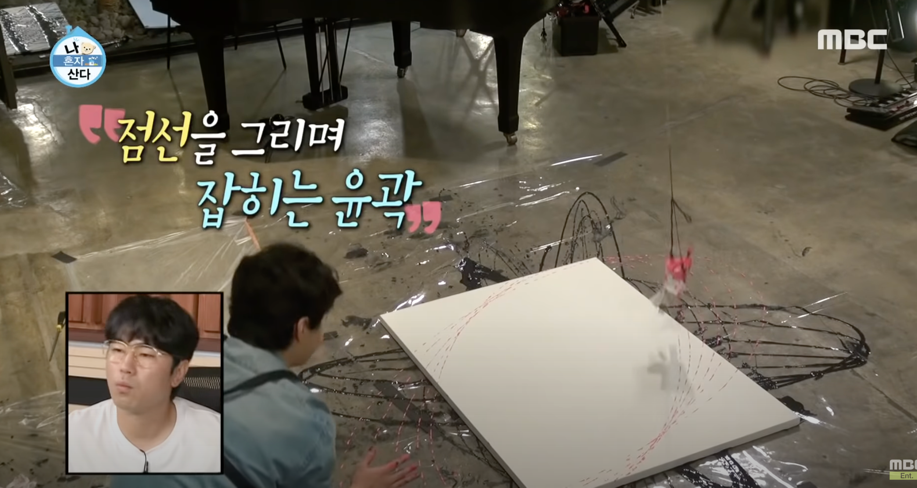
이번 프로젝트의 동기는 유튜브에서 본 두 영상이다.

우선 첫 번째 영상은 개발자이자 유튜버이신 interactive developer이 자신의 포트폴리오를 설명하는 영상이다. 자신의 포트폴리오를 설명하면서, 반 고흐의 그림을 일정 영역으로 나누며, 동그라미로 표현한 프로그램이 있었는데, 이렇게 만든 프로그램이 빈센트 반 고흐 뮤지엄에 직접 전시되었다는 것이 멋있으며, 좋은 아이디어라고 생각했다. 따라서 개인적으로 동그라미에서 더 나아가, 자신이 원하는 모양으로 그림을 만드는 프로그램을 만들어보자는 생각을 갖게 되었다. 아래는 interactive delveloper이 구현한 그림의 캡쳐본이다.



<Interactive developer이 구현한 반 고흐의 그림>

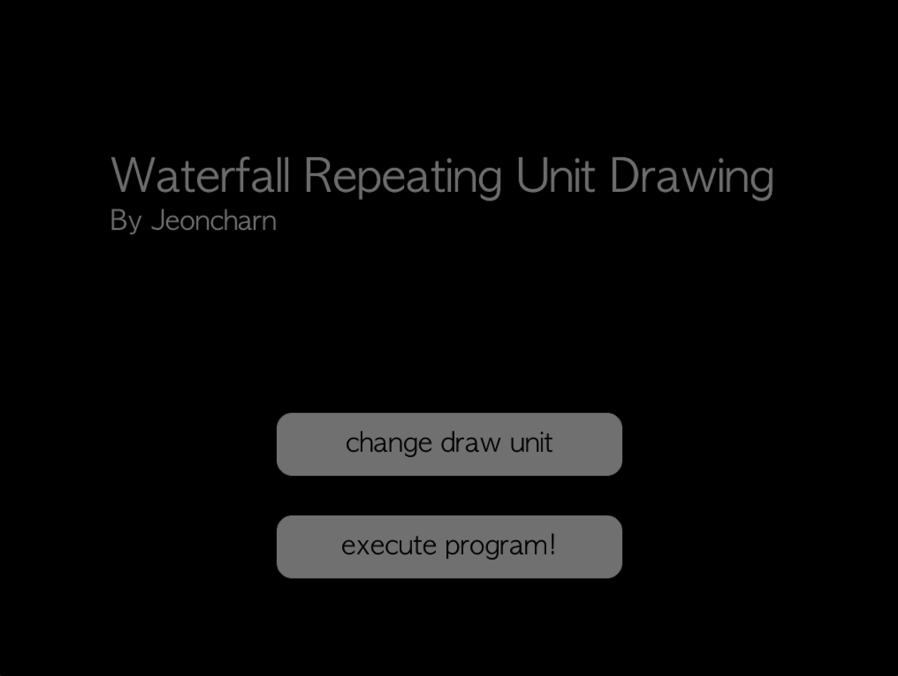
두 번째 영상은 ‘나 혼자 산다’ 에서 헨리가 그림을 그릴 때 사용한 pendulum art 이다. pendulum art란 진자의 운동을 통해서 그림을 그려내는 것인데, 진자에서 물감이 떨어지는 것이 마치 물방울이 떨어지는 것과 비슷하다는 생각을 가지며, 개인적으로 물방울이 떨어져서, 그림이 그려지는 형태를 구현해보면 좋을 것 같다고 생각했다. 아래는 그 영상의 캡쳐본이다.



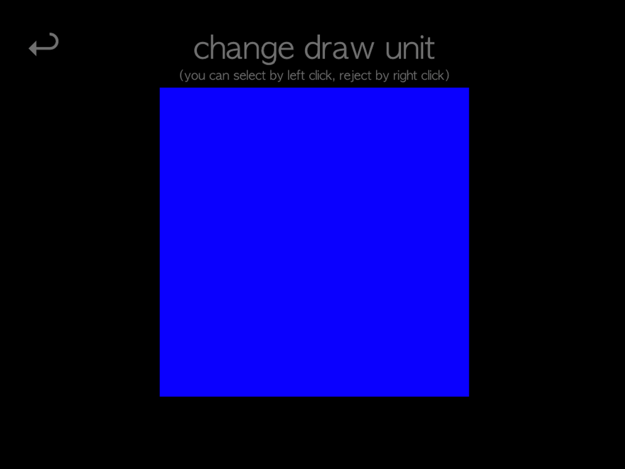
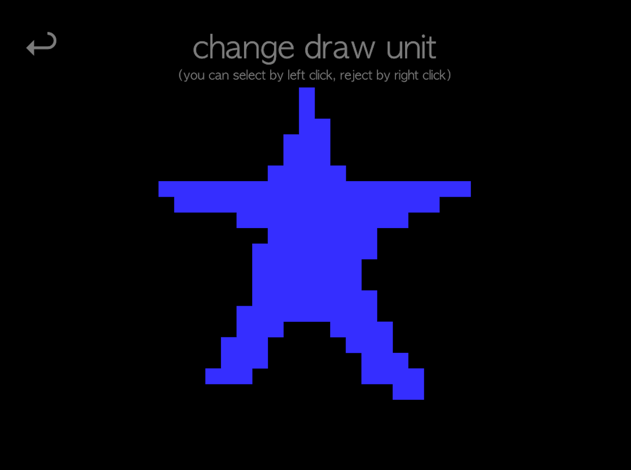
<헨리의 pendulum art>

**6. 프로젝트 실행 결과 캡쳐**

1. 실행한 직후의 main 화면

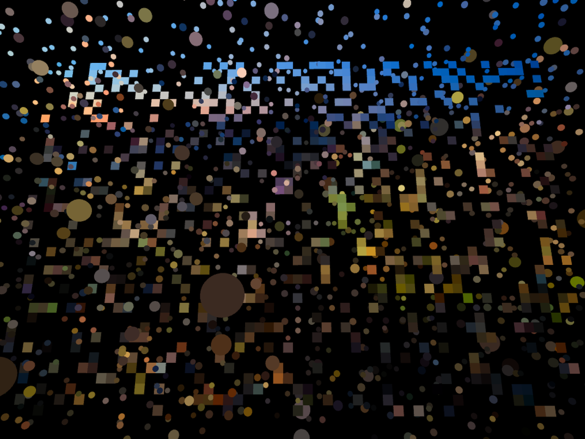


2. 실행 이후 change draw unit 버튼은 눌렀을 때 기본 화면과 unit을 편집한 경우

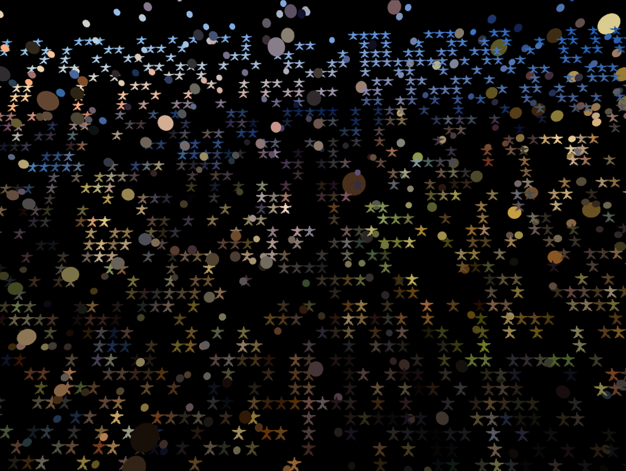
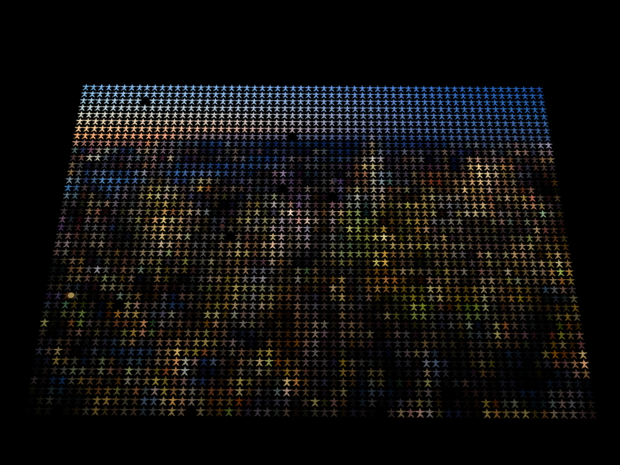
 

3. 각 경우에 대해서 main 화면으로 돌아가 execute program! 을 누른 경우

-편집하지 않은 경우

-편집한 경우

-모든 경우에서 ‘q’키를 입력받은 경우 프로그램이 종료된다.



**7. 느낀 점 및 개선 사항**

사실 이번 프로젝트를 진행하기 전까지만 해도, 학교에서 제시해준 주제의 문제를 풀거나, 정해진 알고리즘을 실행하는 형태만 프로그래밍을 통해서 구현해보았다. 하지만 이번 프로젝트를 통해서, 처음으로 나만의 주제를 정하고 직접 제작해볼 수 있었다. 또한 제작하면서 사실 처음 제작을 목표로 한 프로그램을 제작하다가 개인적인 한계가 존재해서 이번 프로그램으로 바꿔서 제작했는데, 큰 한계에 직면했을 때는, 계속 시도하기보다는 오히려 다른 방법으로 생각했을 때, 더 좋은 결과가 나올 수 있다는 것을 알게 되었다. 또한 이번 프로그램을 제작하면서, 방학 시간에 위 프로그램을 온라인 홈페이지에서 구현해보면 어떨까 하는 생각을 갖기도 했다.

이번 프로젝트의 개선점이 여러 가지가 존재하는데, 첫 번째로는 프로그램 내에서 그림의 input을 받지 못한다는 것이다. waterfall 실습에서 진행했던 input 받기를 이번 프로그램에 적용시켜 보려고 했는데, 제대로 어떻게 하는지 방법을 몰라서 구현하지 못한 아쉬움이 있다. 구글링을 해봐도, data input을 openframeworks 안에서 구현하는 방법을 찾을 수 없었다.

두 번째로는 물방울이 떨어지는 모양이다. 원래 처음 목표는 물방울을 cone + sphere 형태로 구현해주려 했으나, ofDrawCone에서 3차원으로 cone이 보여지는 형태가 구현해놓은 알고리즘의 방향과 맞지 않아서, 이를 포기하고 물방울을 그냥 구의 형태로 구현했다. 시간적 여유가 된다면, 이를 해결해보고 싶다.

마지막으로는 프로그램이 execute된 이후, 3d로 그림과 물방울이 떨어지는 형태를 다양한 시점에서 볼 수 있도록 ofEasyCam을 사용해서 구현했는데, 단지 물방울과 그림만 보여주다 보니, 전체 공간에서 허전함이 느껴졌다. 이를 어떻게 채울지 별다른 아이디어가 떠오르지 않아서, 다른 모양을 추가해주지는 않았다. 나중에 이를 보완해서 3d로 그림과 물방울을 바라볼 때, 더 효과적으로, 또한 더 시각적으로 어떻게 보일지 생각해보고, 구현해보고 싶다.