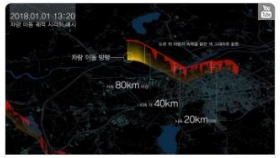
코로나 바이러스로 인한 항공기 운행 감소

제작 방식과 과정

2020.12.

VWL

VWL

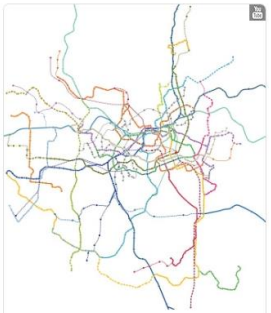


소방차량 시각화 : 속도와 소요시간

글 : 김승범어떻게 하면 GPS데이터로 소방차량의 움직임과 도로의 상황을 효과적으로 보여줄 수 있을까?충남 지역의 소방차량 GPS궤적 데이터를 이용하여 소방출동시 차량이 빠르게 움직이기 어려운 지역을 표현해야 했다. 정지 상태의 이미지가 아닌, 영상을 만들어야 한다는 ....

Fri, 05 Oct 2018

more...

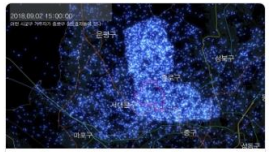


새로 만드는 수도권 지하철은 시간거리를 얼마나 단축시킬까

글 : 김승범 서울을 포함한 수도권은 지하철 건설이 한창이다. GTX 같은 급행철도나 경전철을 포함하여 새로 만드는 노선이 열 개도 넘는다. 이렇게 만들어지는 지하철은 서울과 수도권의 시간거리를 얼마나 단축시킬까? 예전에 한 시간 걸리던 직장을 얼마나 빨리 갈 ....

Mon, 01 Apr 2019

more...

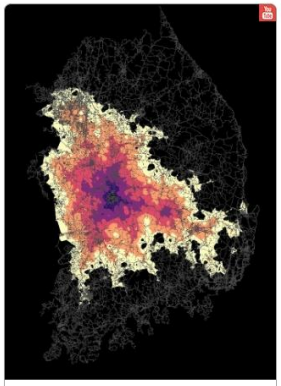


나는 네가 지난 여름에 간 곳을 알고 있다

글 : 김승범글의 제목을 보고 공포 영화를 연상했다면 당신은 20세기에서 온 아재다. 혹은 빅브라더를 연상했다면 안심해도 된다. 우리 부러다는 사실 경찰 당신이 어디 갔는지는 잘 모르기 때문이다. 다만, 예를 들어 당신이 서울 강서구에 산다면, 강서구에 함께 사는 60....

Fri, 07 Dec 2018

more...



실시간 반응형 등시선도(Realtime Interactive Isochrone Map) 그리기

글 : 김승범 몇 단계를 거쳐 <실시간 반응형 등시선도>를 그리는 과정을 설명하려고 한다. 마우스를 지도 위에서 움직이면 해당 지점을 중심으로 다른 곳들이 시간적으로 얼마만큼 걸리는지를 보여주는 지도다.등시선도(isochrone map)라는 용어는 많은 사람들에게 익숙하....

Tue, 26 Mar 2019

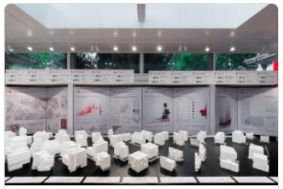
more...

어디서 오고 어디로 가는가 : 인구이동 2015

전국적으로 한 해에 약 600만 건 정도의 이사를 한다2015년 기준으로 전국에는 약 5100만 명의 사람들이 1950만 가구를 이루어 살고 있었다. 그리고 한 해 동안 그 중 약 1070만 명이 거주지를 옮겼다. 가구 전체가 한 번에 이사를 가기도 하고, 가구원 중 ....

Tue, 28 Feb 2017

more...

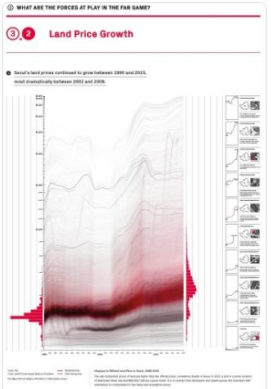


2016 베니스 비엔날레 한국관

2016년 베니스 비엔날레는 전시 총감독인 알레한드로 아라베나(Alejandro Aravena)가 계안한 'Reporting from the Front'라는 주제 아래 5월부터 11월까지 이탈리아 베니스에서 열리고 있다. 전시는 아르세날레의 주제관과 자르디니의 국가관....

Tue, 27 Sep 2016

more...

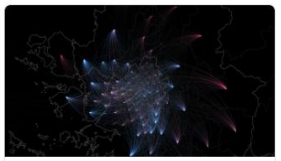


시각화 : 서울, 26년간 공시지가의 변화

들어가며서울 모든 필지의 26년간 공시지가의 변동을 보여주는 위의 패널은 2016년 베니스 비엔날레 한국관 전시를 위해 작업한 것이다. 색상 선정과 패널 레이아웃 작업은 스튜디오 텍스트(http://www.texttexttext.com/)에서 진행하였다. 짧은 글을 통....

Tue, 27 Sep 2016

more...

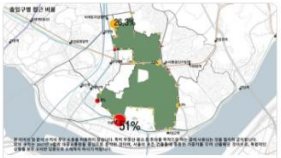


수도권 인구 이동 2012-2016 : 연령에 따른 구분

젊은 사람들은 어디로 움직이는가? 중장년층 이상은 어디로 움직이는가?전체 인구를 40세를 기준으로 나누어 점점 어려질수록 푸른 색으로, 40세에서 점점 나이가 들수록 붉은색으로 표현하여 2012년 1월1일부터 2016년 12월 31일까지의 인구이동을 그려보았다. 시점과....

Sat, 22 Jul 2017

more...

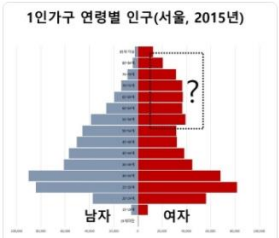


〈용산공원의 바깥도시〉 : 용산공원 라운드테이블 발표자료

2017년 9월 22일 전쟁기념관 이병영홀에서 진행되었던 "용산공원 라운드테이블 1.0"(http://yongsanparkrt.com)에서 발표한 내용 중 주요한 부분들을 옮겨놓았습니다. 용산 공원의 접근성을 중심으로, 주변 지역들의 특징들을 돌아보는 내용입니다.출처를....

Wed, 27 Sep 2017

more...

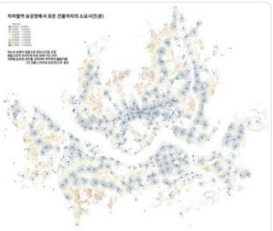


서울 1인가구 분포 : 각자 다른 사정의 사람들

1인가구 연령별 인구 (서울, 2015년)흔히 '1인가구'라고 하면 젊은 남녀들을 떠올린다. 그런데 1인가구의 연령별 그래프를 처음 봤을 때 가장 궁금했던건 55세 이상 여성들이었다. 특이하게도 나이가 들면서 다시 증가한다. 도대체 저 사람은 어떤 사람들일까? 통계청에....

Sat, 15 Apr 2017

more...

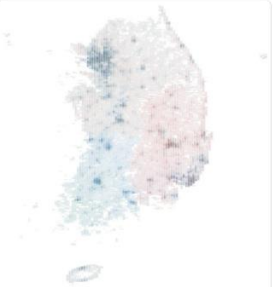


지하철승강장에서 우리집까지는 얼마나 걸릴까

지하철 승강장에서 우리 집, 혹은 회사까지는 얼마나 걸릴까?어떤 역은 승강장이 지하 깊숙하게 있어서 막상 지하철역입구에 들어가서도 한참 걸리고,어떤 집은 평지가 아니라 언덕 꼭대기에 있어서 올라가는데 시간이 오래 걸리며,도중에 버스를 타고 갈 수도 있는데 그러면 시간은....

Tue, 30 May 2017

more...



선거의 득표 수를 어떻게 지도에 표현할 것인가?

선거의 득표 수를 어떻게 지도에 표현할 것인가? 이번 대선 득표 시각화를 준비하면서 시도해보았던 몇 가지 방법들을 기록해본다. 결과적으로 말하자면, 결국 보통의 평범한 지도를 선택하게 되었다.몇 가지 다른 지도들을 만들어 본 배경에는 다음과 같은 세 가지 질문이 있었다....

Mon, 22 May 2017

more...

“주로 공간 데이터를 분석하고 시각화합니다”

작업 방식

Javascript → Processing → C++ / OpenGL

작업 방식 : C++ / OpenGL

처음부터 jpg 이미지 생산까지 모두 코드로 작성

2017년 3월 고위공직자 재산공개 재산 총액

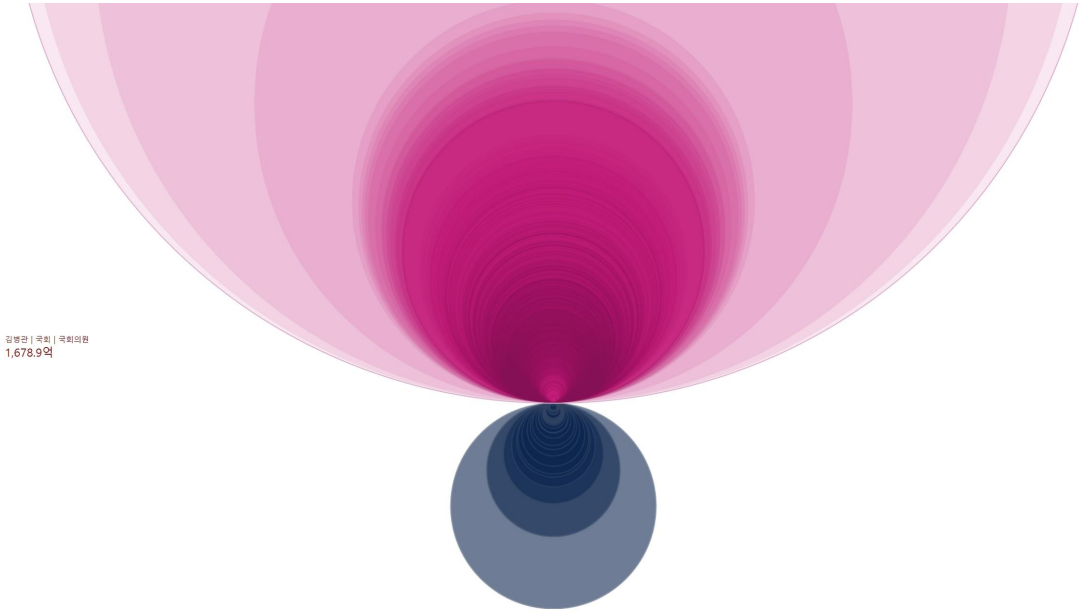
중앙일보 데이터 저널리즘 작업

고위공직자 2천여명의 공직자들의 재산 총액

한 명 당 원 하나

크기는 재산 총액에 비례

마우스를 움직이면 사람과 재산이 표시됨



https://news.joins.com/DigitalSpecial/160

D3.js

2017년 3월 고위공직자 재산공개 재산 총액

중앙일보 데이터 저널리즘 작업

전체적 분포를 보여주면서,

일반적 분포보다 크게 벗어나 있는 것들을 어떻게 강조할 수 있을까?

재산이 많은 사람과 빚이 많은 사람들을 어떻게 대조적으로 보여줄까?

퍼포먼스 저하와 해결

D3.js 에서 SVG 로 원 2300개를 '겹쳐' 그리면

→ 구형 스마트폰에서는 퍼포먼스가 안 나옴

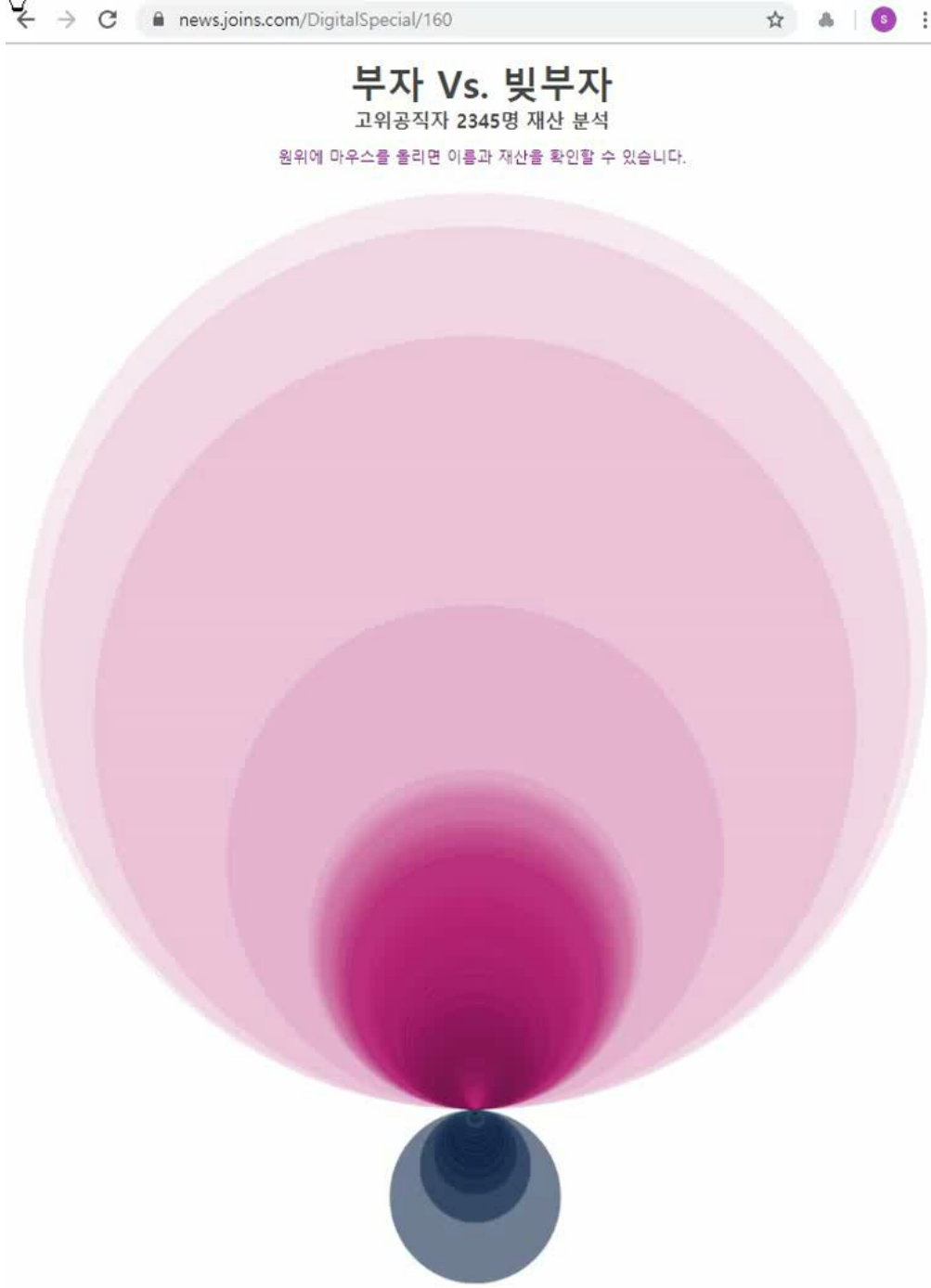
→ 바탕 이미지는 .png 파일로 대체하여 최적화시킴

당시에는 안타깝게도 〈canvas〉의 존재를 알지 못했음

```
for (원의 개수만큼, 작은 원부터) {

  if (마우스가 원 안에 있으면) {
    //하얀 테두리 원 하나, 텍스트 한줄 그린다.
    break;
  }
}
```

https://news.joins.com/DigitalSpecial/160

D3.js

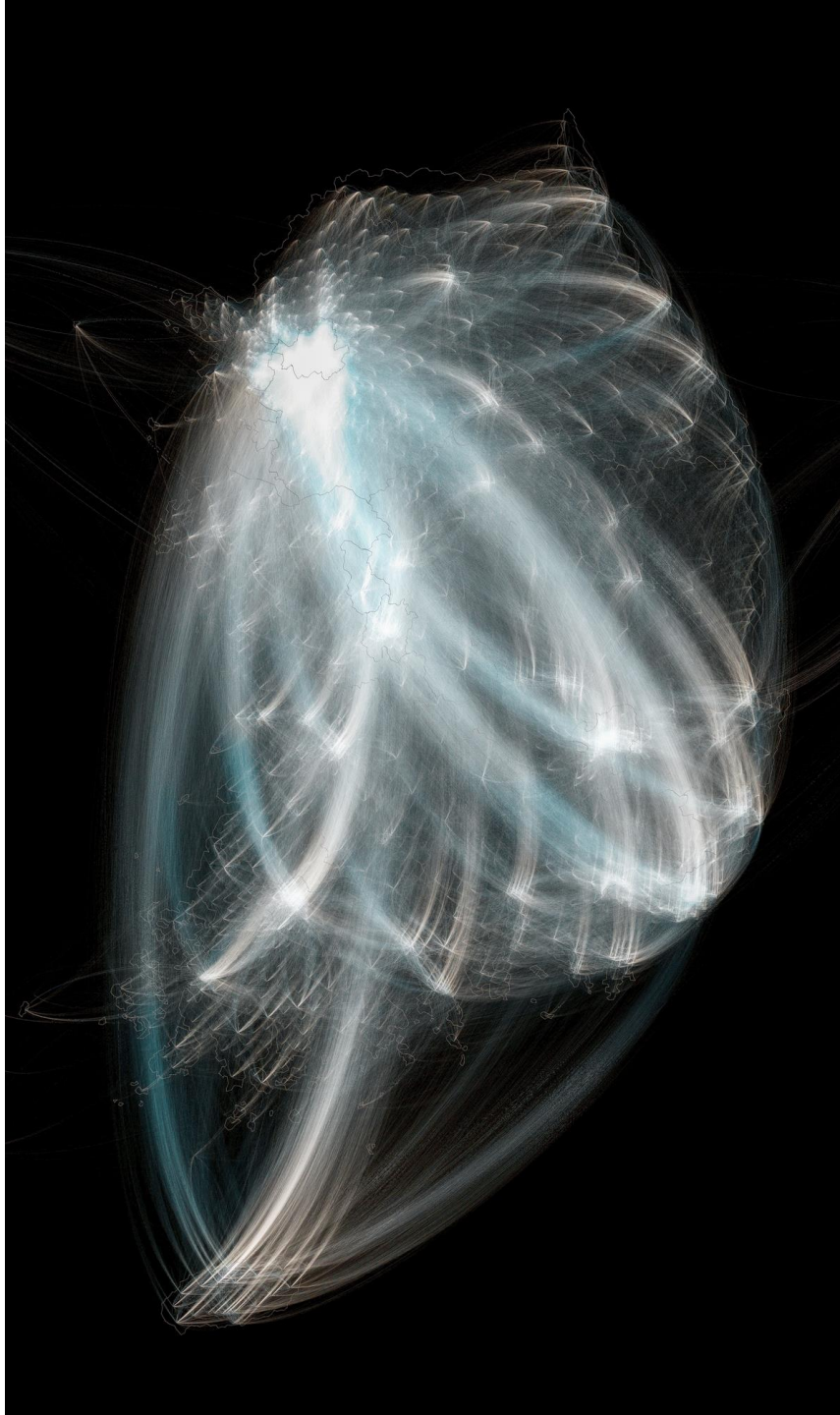


전국 인구이동

580만 가구의 이동 하나에 선 하나씩

Processing 으로 작업

가로 해상도 5000 px 기준 100초 정도 소요



OpenGL로 580만개 선 그리기

한장에 0.03초

→ 1초에 30프레임

작업 방식 : C++ / OpenGL

단점

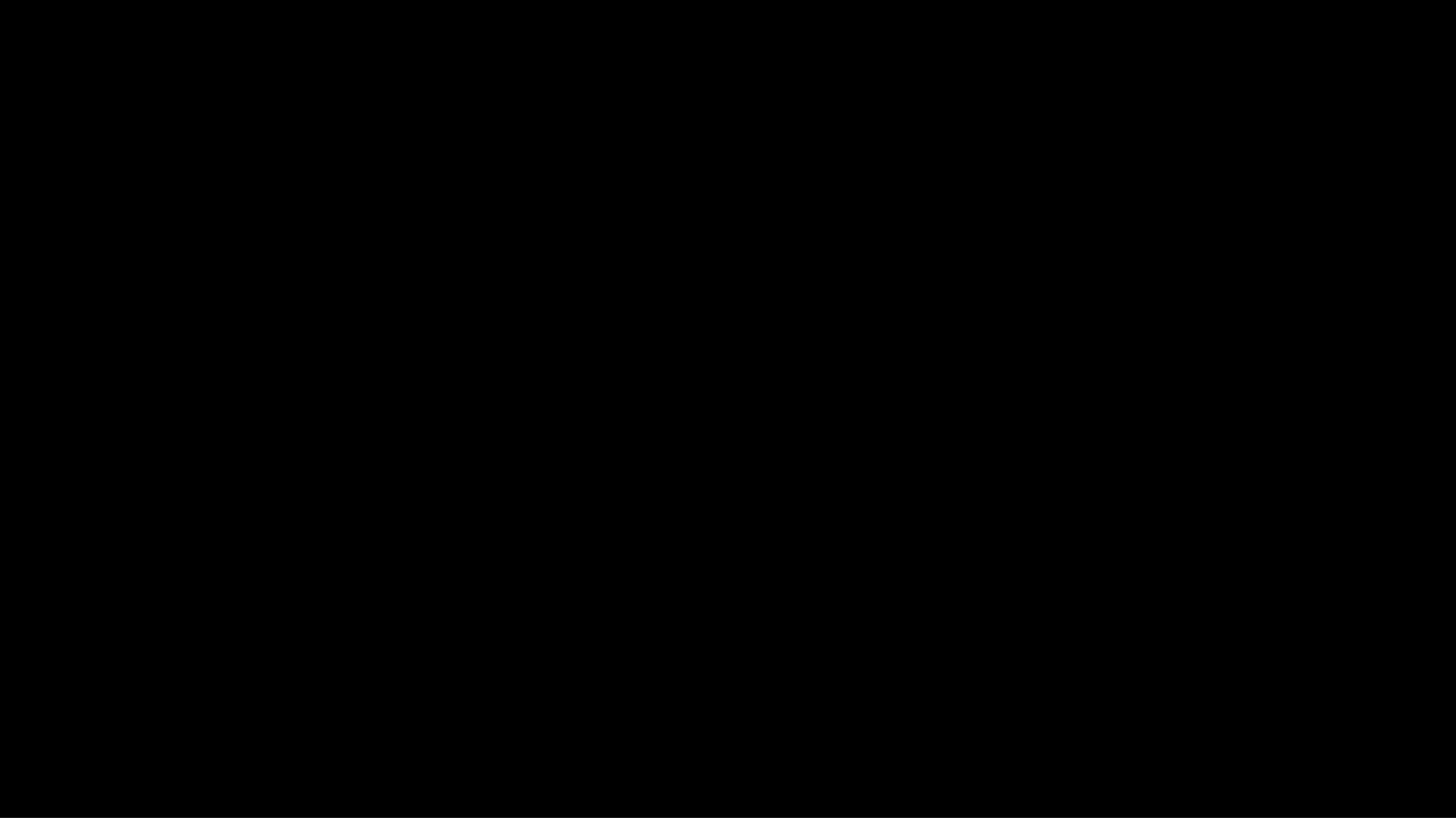
생산성이 매우 낮다
(web으로) 소통하기 어렵다

장점

performance 가 좋다

제작 과정

코로나 바이러스로 인한 항공기 운행 감소



코로나 바이러스로 인한 항공기 운행 감소

전시 영상

제작 과정

지구를 만들자

작업 방식 : C++ / OpenGL

처음부터 jpg 이미지 생산까지 모두 코드로 작성

구를 그린다

↓

낮과 밤의 이미지를 맵핑

↓

대기를 만든다

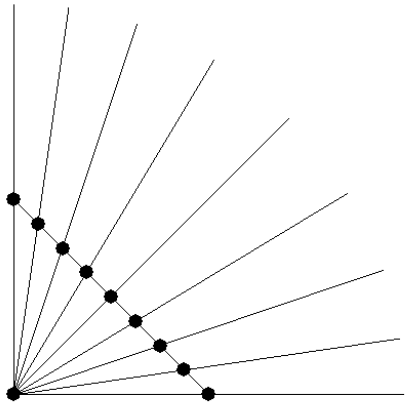
↓

계절에 따른 태양의 위치, 시선의 위치를 고려해서
밝은 곳과 어두운 곳을 만든다

시간 입력값과 시점 조정에 따라
정확히 계산되도록 설정

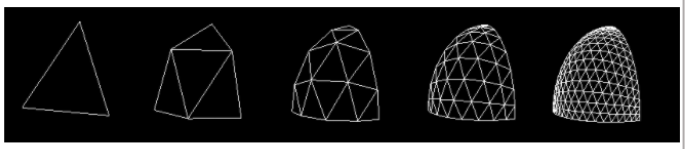
구를 그리는 방법 : 재귀적 호출을 통한 삼각형의 분해

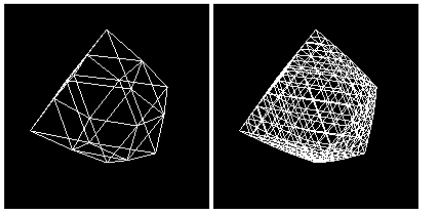
If we do this normalization process on a lot of points, all with respect to the same point A and with the same distance R, then the normalized points will all lie on the arc of a circle with center A and radius R.



Here, the black points begin on a line and "bulge out" into an arc.

This process can be extended into three dimensions, in which case you get a sphere rather than a circle. Just add a dz component to the normalize function.

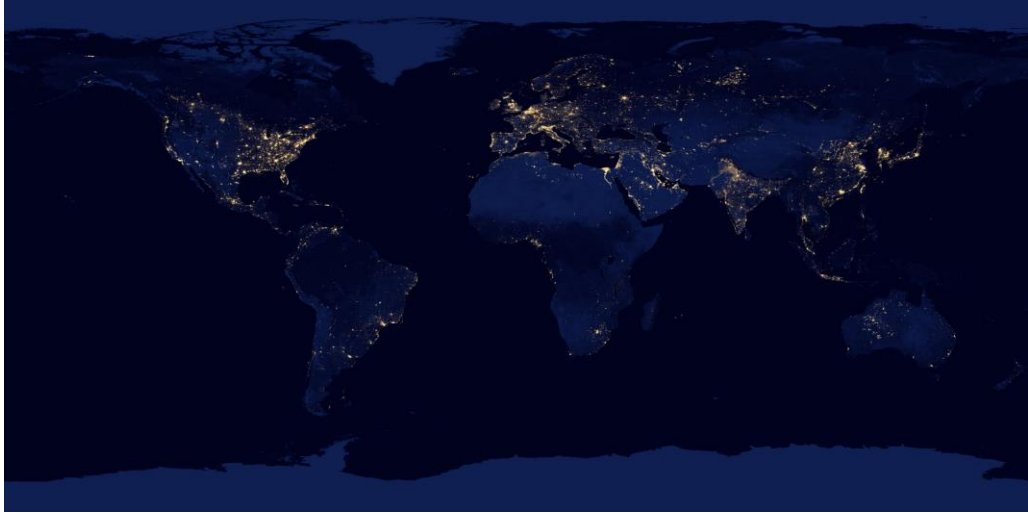




https://stackoverflow.com/questions/7687148/drawing-sphere-in-opengl-without-using-glusphere

```
void makeEarthSphere(GLuint& vao, GLuint& vertexSize)
{
	GLint drawLevel;
	if (isFinal) drawLevel = 7;
	else drawLevel = 4;
	indexSphere = 0;
	subdivide(-1.0, -1.0, 0.0, 1.0, -1.0, 0.0, 0.0, 0.0, 1.0, drawLevel);
	subdivide(1.0, -1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, drawLevel);
	subdivide(1.0, 1.0, 0.0, -1.0, 1.0, 0.0, 0.0, 0.0, 1.0, drawLevel);
	subdivide(-1.0, 1.0, 0.0, -1.0, -1.0, 0.0, 0.0, 0.0, 1.0, drawLevel);
	subdivide(-1.0, -1.0, 0.0, 1.0, -1.0, 0.0, 0.0, 0.0, -1.0, drawLevel);
	subdivide(1.0, -1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, -1.0, drawLevel);
	subdivide(1.0, 1.0, 0.0, -1.0, 1.0, 0.0, 0.0, 0.0, -1.0, drawLevel);
	subdivide(-1.0, 1.0, 0.0, -1.0, -1.0, 0.0, 0.0, 0.0, -1.0, drawLevel);

}
```

```
void subdivide(float v1x, float v1y, float v1z,  float v2x, float v2y, float v2z, float v3x, float v3y, float v3z, int level) {

	if (level == 0) {
		float s1 = sqrt(v1x * v1x + v1y * v1y + v1z * v1z);
		float s2 = sqrt(v2x * v2x + v2y * v2y + v2z * v2z);
		float s3 = sqrt(v3x * v3x + v3y * v3y + v3z * v3z);

		vertex_position.push_back(vec3(v1x / s1, v1y / s1, v1z / s1));
		vertex_position.push_back(vec3(v2x / s2, v2y / s2, v2z / s2));
		vertex_position.push_back(vec3(v3x / s3, v3y / s3, v3z / s3));
		vertex_index.push_back(indexSphere++);
		vertex_index.push_back(indexSphere++);
		vertex_index.push_back(indexSphere++);
	}
	else {
		float v12x = 0.5f * (v1x + v2x);
		float v12y = 0.5f * (v1y + v2y);
		float v12z = 0.5f * (v1z + v2z);

		float v13x = 0.5f * (v1x + v3x);
		float v13y = 0.5f * (v1y + v3y);
		float v13z = 0.5f * (v1z + v3z);

		float v23x = 0.5f * (v2x + v3x);
		float v23y = 0.5f * (v2y + v3y);
		float v23z = 0.5f * (v2z + v3z);

		subdivide(v1x, v1y, v1z, v12x, v12y, v12z, v13x, v13y, v13z, level - 1);
		subdivide(v12x, v12y, v12z, v2x, v2y, v2z, v23x, v23y, v23z, level - 1);
		subdivide(v13x, v13y, v13z, v23x, v23y, v23z, v3x, v3y, v3z, level - 1);
		subdivide(v12x, v12y, v12z, v23x, v23y, v23z, v13x, v13y, v13z, level - 1);
	}
}
```



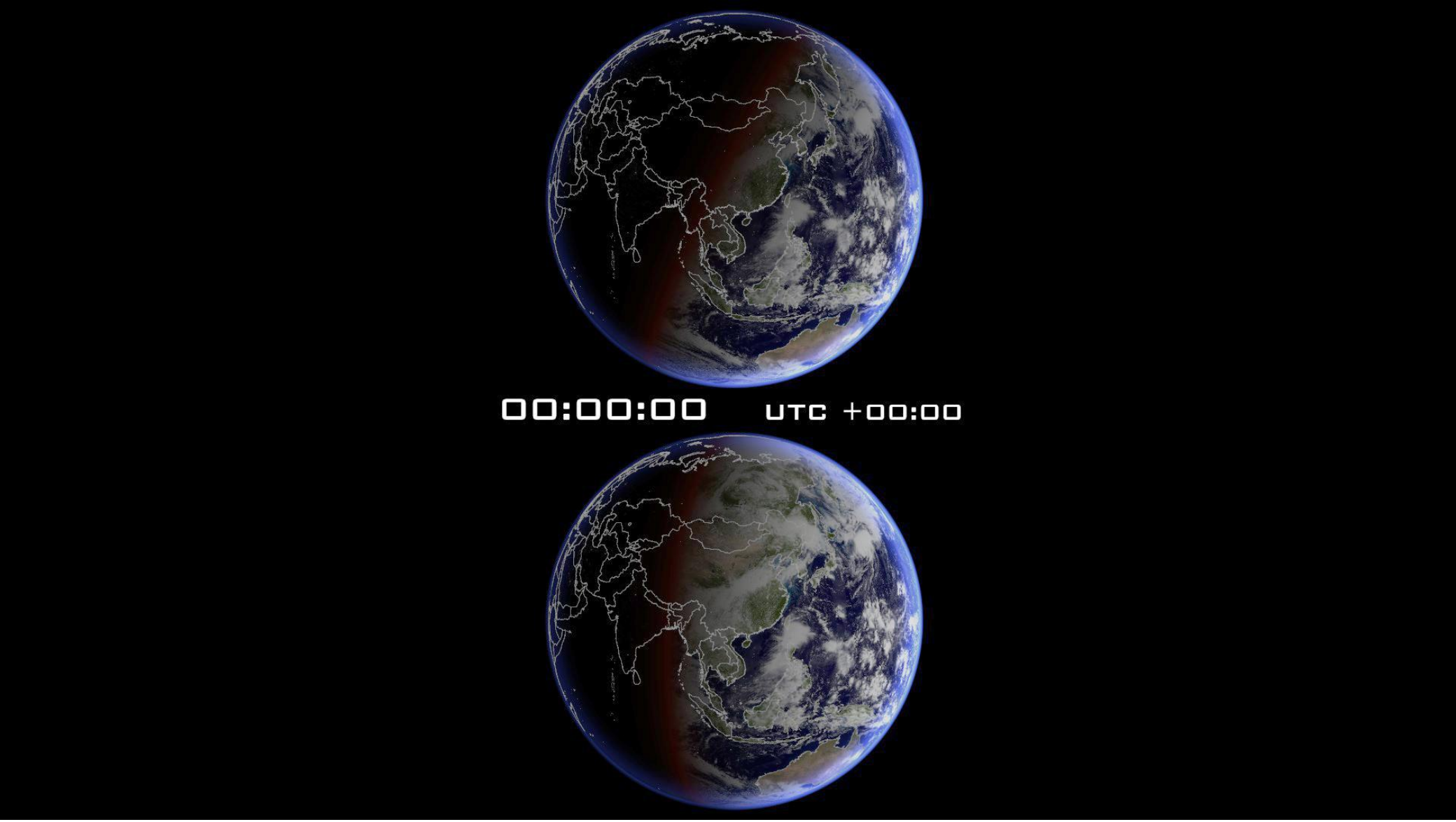
이미지 출처 : NASA

각각의 이미지는 경위도 좌표가 매핑을 위해 최적화되어 있는 정교한 이미지

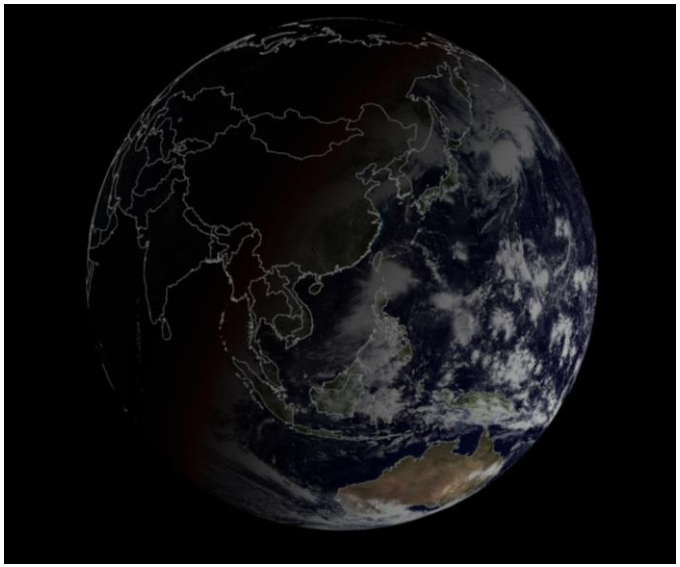
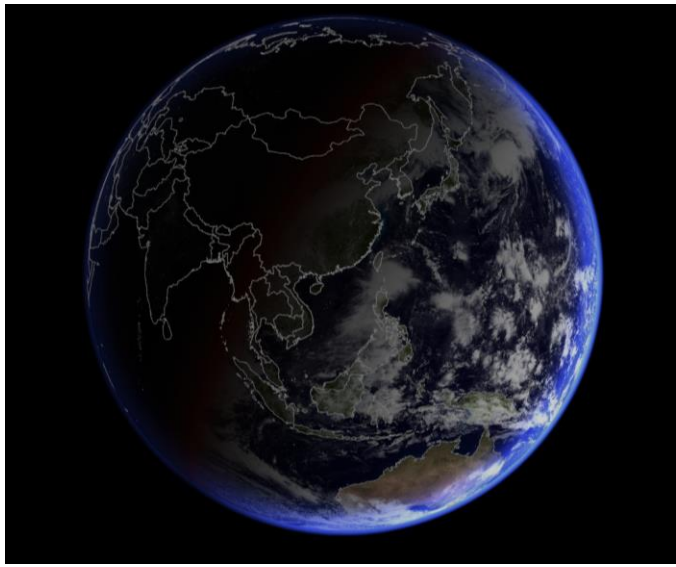
→ 태양의 위치에 따라 낮과 밤을 계산해서

→ 밤에는 밤의 픽셀을, 낮에는 낮의 픽셀을, 낮과 밤이 만나는 곳은 여명을 표현

```
vec3 ColorDayNight(vec3 objectC1, vec3 objectC2, vec3 normal) {
       return 1.0 *(objectC1 *(max(0.0,dot(lightDirection, normal)))
                    + objectC2  * (max(0.0,dot(-lightDirection, normal)))
                    + vec3(1,0.25,0.1) * max(0.0, pow( 0.80* (1.0- abs(dot(lightDirection,normal)))  ,12.0)) );
}

vec3 dayPic = texture2D(dayTexture, longitudeLatitude).rgb;       //낮의 이미지
vec3 nightPic = texture2D(nightTexture, longitudeLatitude).rgb;   //밤의 이미지

vec4 theColor = vec4(ColorDayNight(dayPic,nightPic, normal),1.0); //함수 호출을 통해 낮과 밤을 섞는다
```



같은 시각이라도 1월과 3월의 낮과 밤은 다름

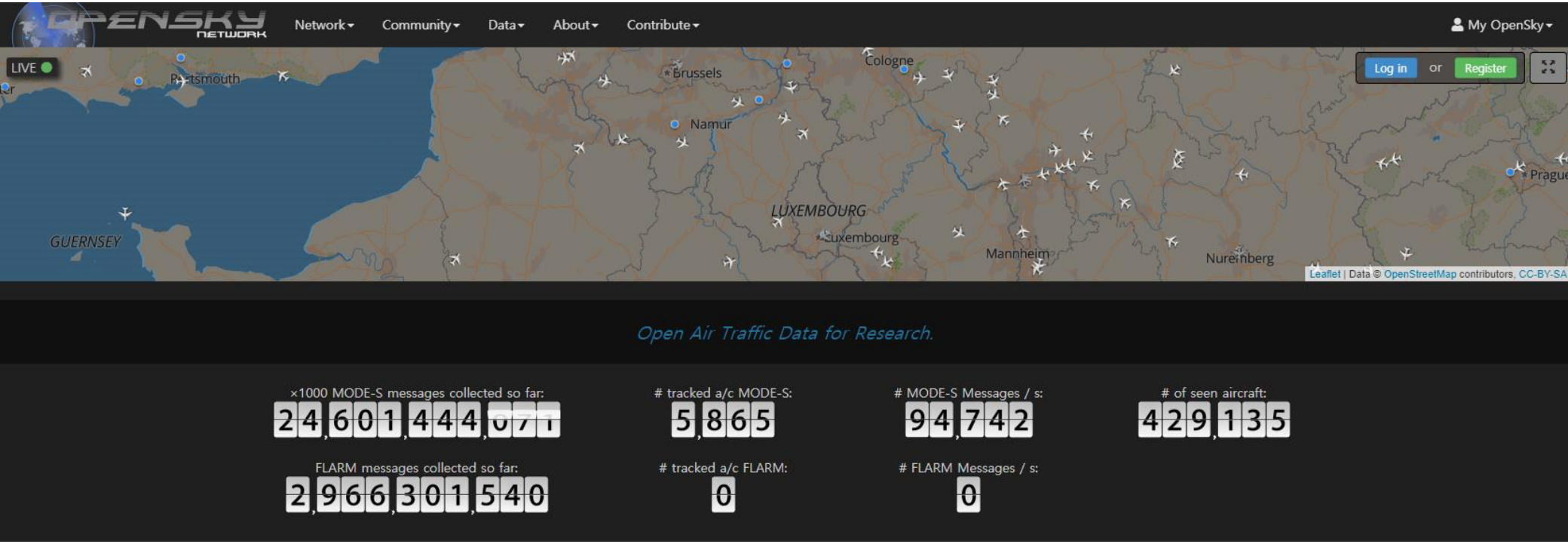
지구 반지름 6378km 밖으로 100km 의 대기권을 설정한 후 대기 효과 계산

```
vec3 calculateRim(vec3 View, vec3 Normal)
{
	float f = 1.0 -abs(dot(Normal,View));

	f = max(0.0,smoothstep(0.4,0.8,f)-pow(f,rim_power)*2);
	f = pow(f,2)*2;
	//f = pow(f*2.0-1.3, rim_power)-pow(f*2.0-1.3, rim_power-2.0);
	//f = f*2;
	//f = -(f*8-6)*(f*8-6)+1;
	//f = smoothstep(0.0,0.9,f);
	//f = 1.0-pow(1.0-f, 30);
	//f = smoothstep(0.0,1.0,f);
	return f*rim_color;
}
```

지구 완성!

제작 과정

항공기 데이터 처리



data source： opensky-network.org

항공 교통 상호 관제를 위해 1초 단위로 broadcast 하는 1090MHz 채널의 정보를 수집하여 공개

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	time	icao24	lat	lon	velocity	heading	vertrate	callsign	onground	alert	spi	squawk	baroaltitude	geoaltitude	lastposupdate	lastcontact
2	1586131200	a546c9	38.791560	-77.0839843	136.409170153	326.8493119	-14.30528000	RPA4354	False	False	False	3771	2034.5400000000002	2057.4	1586131198.944	1586131200.0
3	1586131200	aa147a	39.997741	-75.2186279	199.100462217	243.01063562	-13.32992000	FDX990	False	False	False	5635	4015.7400000000002	4015.7400000000002	1586131198.792	1586131200.0
4	1586131200	c040e8	43.162117	-76.9650486	255.164224000	90.0	0.0	WSW114	False	False	False	0501	12496.800000000001	12397.740000000002	1586131198.925	1586131200.0
5	1586131200	a10e29	39.668563	-76.4462147	209.540436280	230.8788476	-0.32512	UAL241	False	False	False	3320	12192.0	12192.0	1586131198.631	1586131200.0
6	1586131200	a86605	39.203613	-75.6513180	233.074927694	38.81781069	-4.55168	JBU1534	False	False	False	7413	11490.960000000001	11521.44	1586131198.968	1586131200.0
7	1586131200	aaaa1e	39.623474	-75.3712264	178.016208434	180.8279160	21.45792	SWA3641	False	False	False	1652	3726.1800000000003	3733.8	1586131198.068	1586131200.0
8	1586131200	a87196	39.940063	-77.0090942	217.315297220	272.5779268	11.3792	NKS395	False	False	False	3341	9220.2	9250.68	1586131198.631	1586131200.0
9	1586131200	a5fff9	50.750015	4.863454457	178.553574505	292.3531976	6.5024000000	ELY841	False	False	False	0105	3406.1400000000003	3497.58	1586131198.938	1586131200.0
10	1586131200	a62c1a	38.791467	-77.4445190	183.159403988	276.1269330	19.832320000	SWA4849	False	False	False	3611	5791.200000000001	5836.920000000001	1586131198.67	1586131200.0
11	1586131210	a051c9	33.774169	-111.733311	70.5088621261	237.8173682	-1.30048	N12EM	False	False	True	5406	1341.1200000000001	1295.4	1586131209.687	1586131209.955
12	1586131210	a8233a	42.399215	-80.3182359	262.915110922	86.29778714	-0.32512	ASQ4326	False	False	False	6523	11277.6	11201.4	1586131209.991	1586131209.991
13	1586131210	a3e2ea	34.751524	-110.193901	144.250868163	256.8062807	0.0	N35ET	False	False	False	1342	12192.0	12359.64	1586131209.564	1586131209.943
14	1586131210	a87196	39.941024	-77.0368652	216.287455435	272.5901859	7.1526400000	NKS395	False	False	False	3341	9319.26	9349.74	1586131209.64	1586131209.99
15	1586131210	78158a	44.087576	-1.10648018	257.297097861	334.73862528	-0.32512	DKH1640	False	False	False	2313	10972.800000000001	11102.340000000002	1586131209.953	1586131209.982
16	1586131210	acb9be	37.375441	-118.419242	160.784943695	247.6198649	-3.251200000	N919SV	False	False	False	6001	10523.220000000001	10370.820000000002	1586131209.588	1586131209.588
17	1586131210	a973e0	32.740173	-114.443041	168.393796208	183.1572265	6.8275200000	SKW3100	False	False	False		3307.0800000000004		1586131209.793	1586131209.793
18	1586131210	a93b12	33.887878	-112.068282	71.5151176888	179.1756571	0.0	N694RJ	False	False	False	6157	1417.3200000000002	1356.3600000000001	1586131209.907	1586131209.954

결측값을 포함한 정제되지 않은 데이터가 뒤섞여 있음

→ 항공기 고유 코드를 기반으로 취합 후 데이터 정제

```
if (temp[2].equals("")||temp[3].equals("")) continue; //위경도 중 하나라도 없으면 건너뜀
String lat = temp[3];
String lon = temp[2];

String altitude;
if (!temp[13].equals("")) altitude = temp[13]; //기하고도로 저장 우선권
else if (!temp[12].equals("")) altitude = temp[12]; //없으면 기압고도로 저장. 데이터 중에 기압고도만 있는 경우가 꽤 있음
else altitude = "-9999"; //고도가 모두 없을 경우 특별히 -999로 표시함

double timeRecord;
if (!temp[14].equals("")) timeRecord = Double.parseDouble(temp[14]);
else timeRecord =  Double.parseDouble(temp[15]);    //실제 측정치가 비어 있을 경우 최종 컨택시점을 저장
double paperRecord = Double.parseDouble(temp[0]);

double timegap = paperRecord-timeRecord;
if (timegap>300) continue; //마지막으로 받은지 300초 이상 지났으면 자료 폐기

double velocity; //속도는 항공기 한번 비행을 split하기 위해 필요함
if (temp[4].equals("")) velocity = -999;
else velocity = Double.parseDouble(temp[4]);

String time = ""+((int)Math.round(timeRecord));

String value = id +","+ time  +","+ lon +"," + lat +"," + altitude +","+velocity;
```

```
//이제 선형 보간. 속도와 고도는 선형보간하면 되는데 lat lon은 회전을 고려해야 한다.
int t1 = t - previous.time;
int t2 = ap.time-t;

float lon, lat, altitude, velocity;
Vec3 xyz;

if ((t1+t2)==0) {
    lon = ap.lon;
    lat = ap.lat;
    altitude = ap.altitude;
    velocity = ap.velocity;
    xyz = geodeticToCartesian(lon, lat, altitude);
} else if (previous.lon==ap.lon && previous.lat==ap.lat){
    lon = ap.lon;
    lat = ap.lat;
    altitude = ap.altitude;
    velocity = ap.velocity;
    xyz = geodeticToCartesian(lon, lat, altitude);

} else {

    Vec3 vecPre = geodeticToCartesianNormal(previous.lon, previous.lat);
    Vec3 vecJ = geodeticToCartesianNormal(ap.lon, ap.lat);
    Vec3 normal1 = crossProduct(vecPre, vecJ);
    double dot1 = dotProduct(vecPre, vecJ);
    double rad = Math.acos(dot1);

    Vec3 rotatedVec = rotateVectorCC(vecPre, normal1, (rad/(t1+t2) * t1));
    altitude = (t1*ap.altitude + t2*previous.altitude)/(t1+t2);
    velocity = (t1*ap.velocity + t2*previous.velocity)/(t1+t2);
    double newLen = altitude + WGS84_EQUATORIAL_RADIUS;
    xyz = new Vec3(newLen*rotatedVec.x,  newLen*rotatedVec.y, newLen*rotatedVec.z);
```

여러 차례 처리의 시행착오를 통해

유효한 값과 결측값의 정도를 파악한 후

데이터를 손상시키지 않는 범위에서

'시각화를 위해' 보정

항공기의 착륙 후 다음 비행 전에 세션을 끊고,

대서양 등 센서가 없어서 데이터가 누락된 지역은 보간

이상치를 탐지하여 수정하거나 제거

……

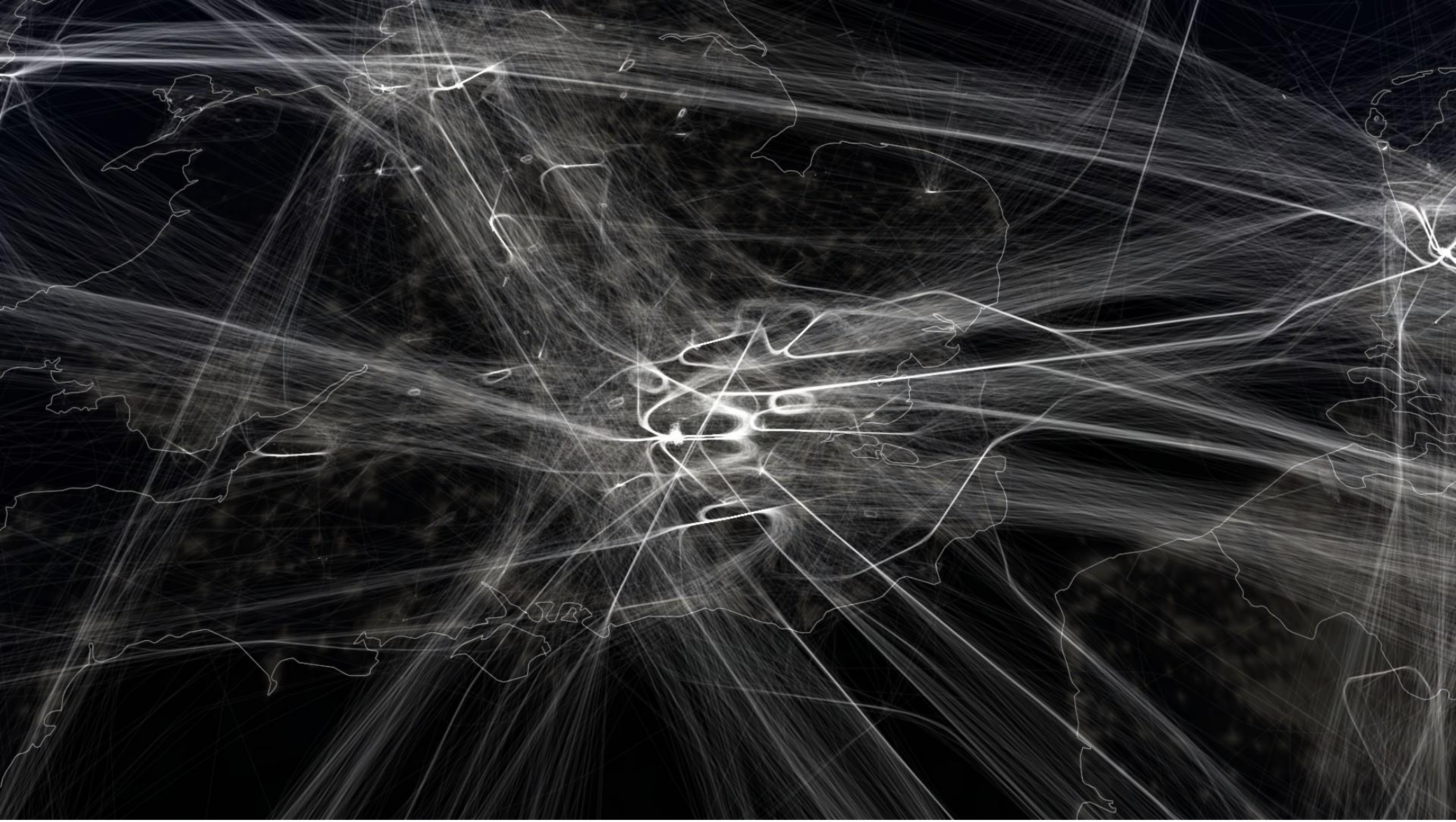
	0	1	2	3	4	5
1	id	time	x	y	z	velocity
2	000001	52380	1102972.1107191637	-4836893.091444052	4009090.756978207	749.268127
3	000001	52440	1104671.7264719203	-4836971.7463397505	4008433.069048565	356.265217
4	000001	52500	1105432.3345831637	-4838977.003515542	4005826.543222767	58.38039
5	000001	52560	1106831.4345092801	-4840775.985598087	4003327.651979718	57.90221
6	000001	52620	1109960.1026806969	-4840962.274929566	4002159.832286179	57.012074
7	000001	52680	1113649.980694032	-4840049.451665479	4002216.734044058	67.89881
8	000001	52740	1114168.117791754	-4839953.437240139	4002175.559219437	66.658844
9	000001	52800	1115789.528281355	-4839937.546887958	4001402.593114337	20.321789
10	000001	52860	1116073.3070286643	-4839791.511697819	4001412.346349071	9.281806
11	000001	53760	1112223.9900073502	-4836419.072964499	4006829.322807839	51.879833
12	000001	53820	1112968.7973894537	-4834470.224624268	4009240.516877790	450.165543
13	000001	53880	1111046.1501520576	-4834026.23036353	4010357.478776691	550.418137
14	000001	53940	1109199.87432936	-4835826.464634978	4008637.206577601	57.803455
15	000001	54000	1107631.6006720832	-4834498.851873224	4010690.460729934	650.744804
16	000001	54060	1106986.3158527666	-4832747.288285306	4012970.24183371	50.651302
17	000001	54120	1105859.1892623154	-4831126.160580732	4015246.455796877	48.000294
18	000001	54180	1104452.7654955222	-4829865.495566774	4017125.538834625	448.447964
19	000001	54240	1107069.8296553746	-4829454.198903605	4016911.743102501	654.738003
20	000001	54300	1110153.620408243	-4829417.20882774	4016118.253470362	650.71246
21	000001	54360	1113196.5326090162	-4828919.154653976	4015862.848406984	552.38279
22	000001	54420	1116127.191658728	-4828212.274269129	4015934.432892614	51.961388
23	000001	54480	1119131.7763165056	-4827376.368256007	4016115.252035552	51.04155
24	000001	54540	1122095.8340480956	-4827072.52917753	4015534.349560342	53.099346
25	000001	54600	1124417.0824366766	-4828051.011698291	4013718.355195753	352.862137
26	000001	54660	1126518.5107772807	-4829386.054042625	4011510.259251071	656.73378

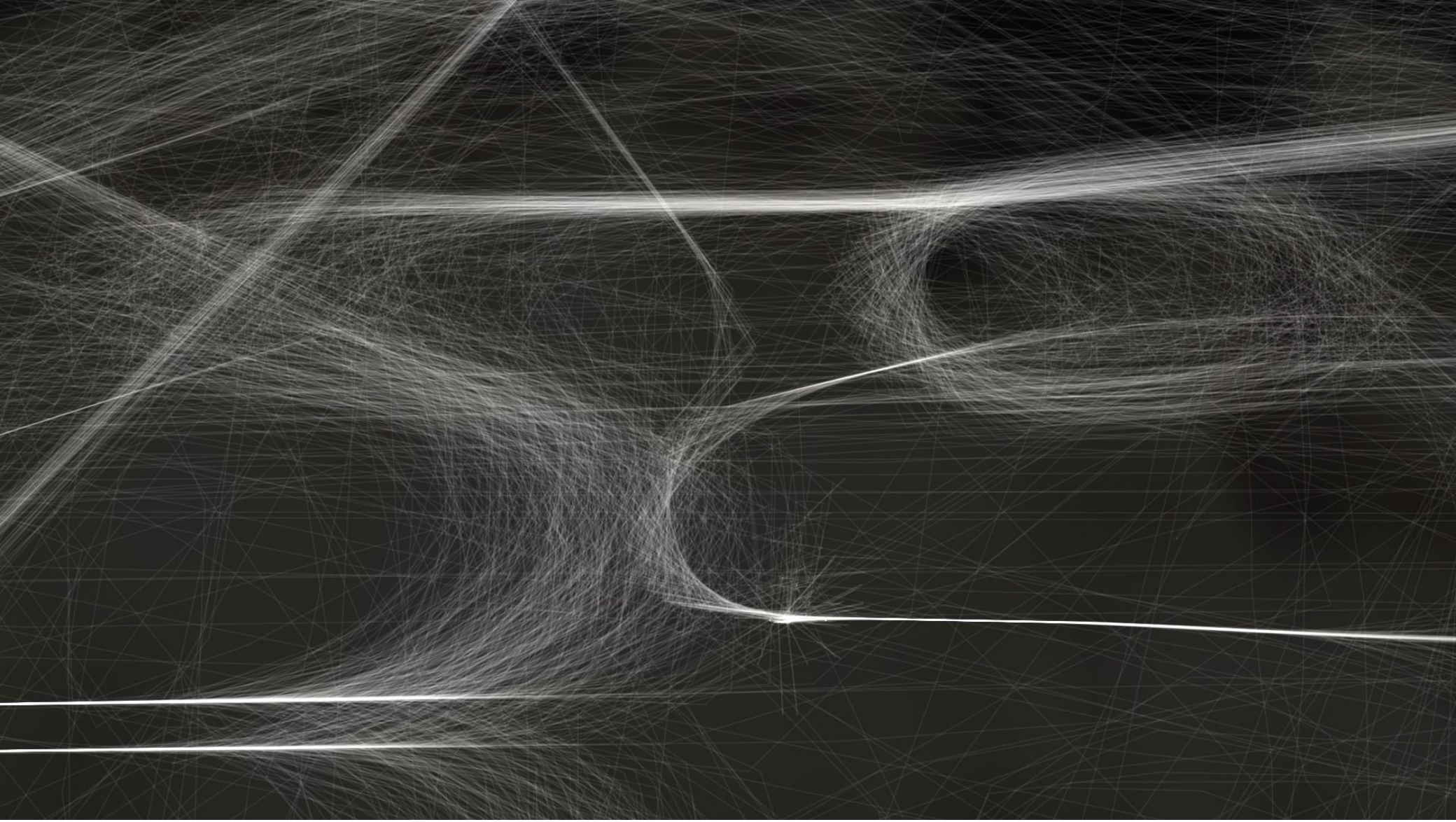
시각화에 필요한

시간과 공간좌표정보만 남기는 방향으로 정제.

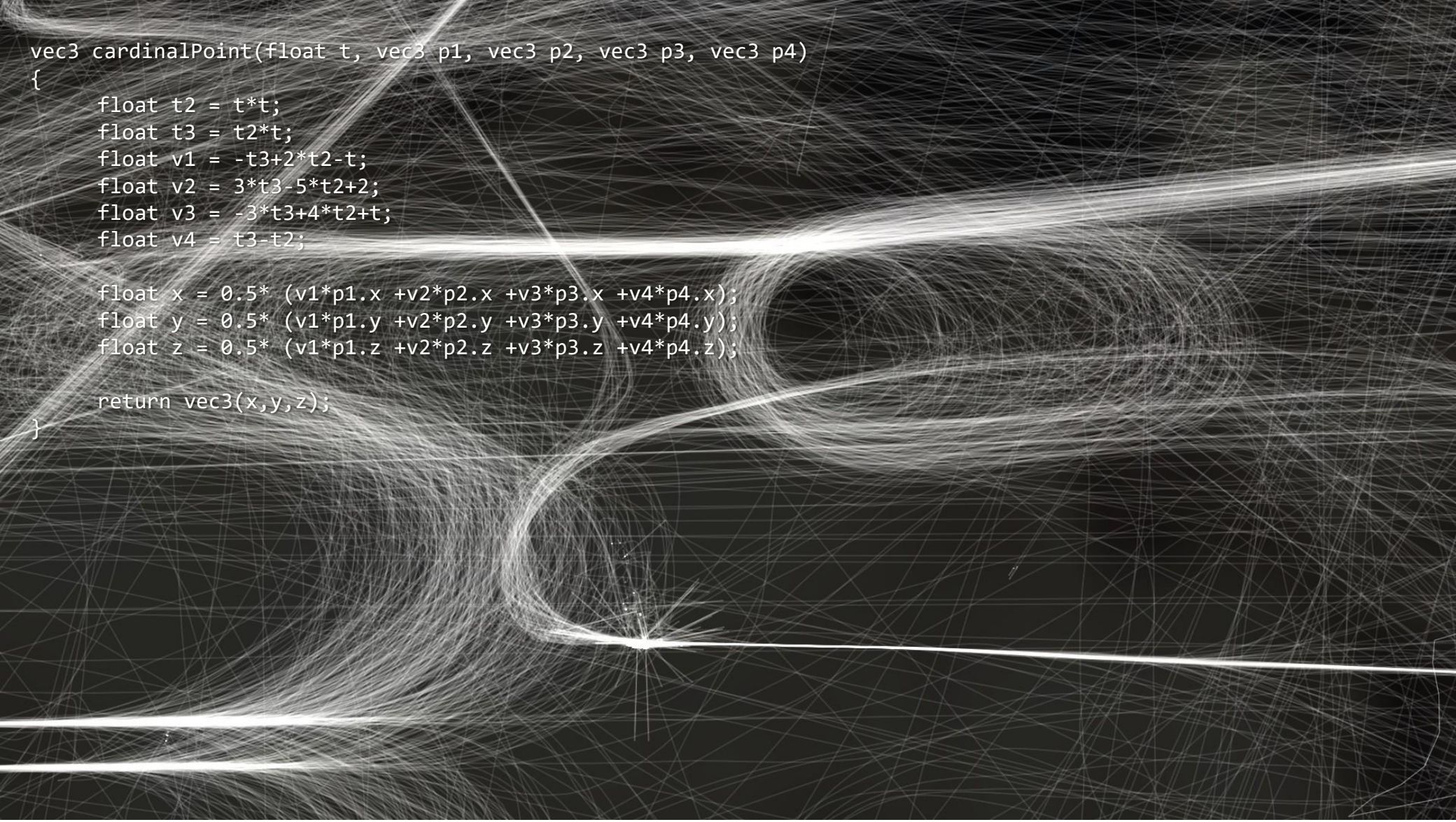
제작 과정

항공기 궤적 그리기





시간간격이 있는 데이터이므로 꺾인 직선들로 표현됨

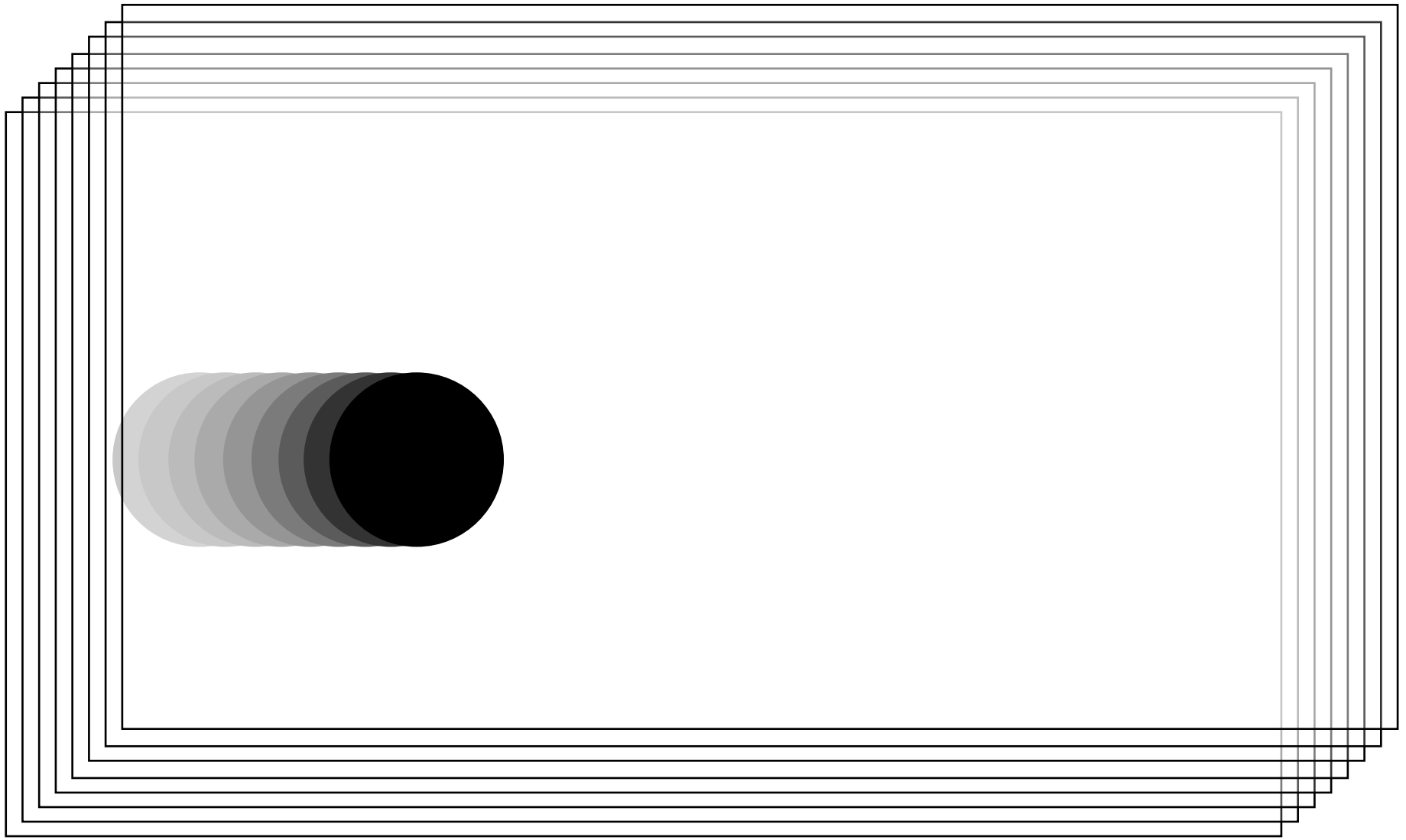


```
vec3 cardinalPoint(float t, vec3 p1, vec3 p2, vec3 p3, vec3 p4)
{
	float t2 = t*t;
	float t3 = t2*t;
	float v1 = -t3+2*t2-t;
	float v2 = 3*t3-5*t2+2;
	float v3 = -3*t3+4*t2+t;
	float v4 = t3-t2;

	float x = 0.5* (v1*p1.x +v2*p2.x +v3*p3.x +v4*p4.x);
	float y = 0.5* (v1*p1.y +v2*p2.y +v3*p3.y +v4*p4.y);
	float z = 0.5* (v1*p1.z +v2*p2.z +v3*p3.z +v4*p4.z);

	return vec3(x,y,z);
}
```

직선 경로를 곡선으로 변형

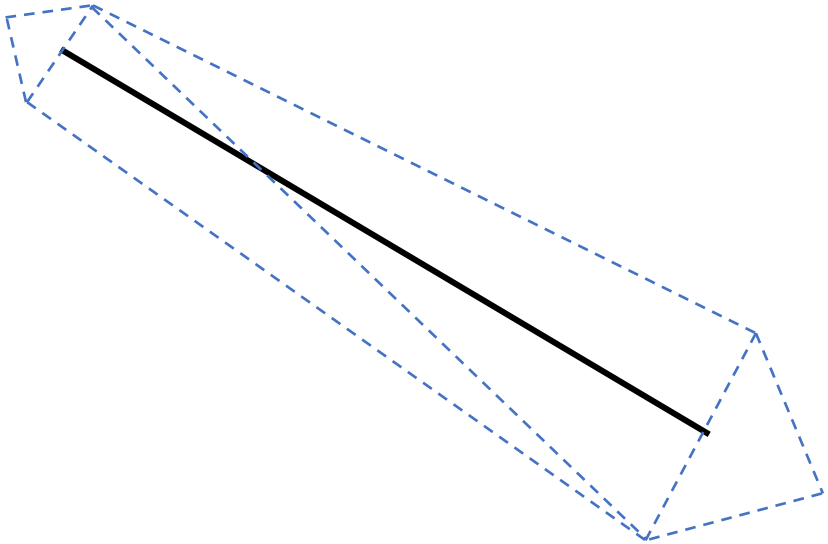
꼬리를 표현

한 프레임에 전체 궤적을 그린 후 → 모두 지우고 → 다음 프레임을 그림



[비교] 정지된 시점의 애니메이션 → 그린 후 반투명한 개체를 덮어씌우는 방식으로 이동을 표현

밋밋하므로 머리 부분 추가



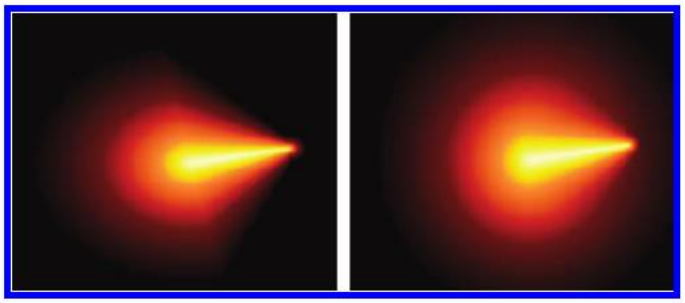


Figure 11.5. The visual error resulting from using the vertex shader–based extrusion while viewing a line along its direction (left). Corrected version using geometry shader–based extrusion (right).

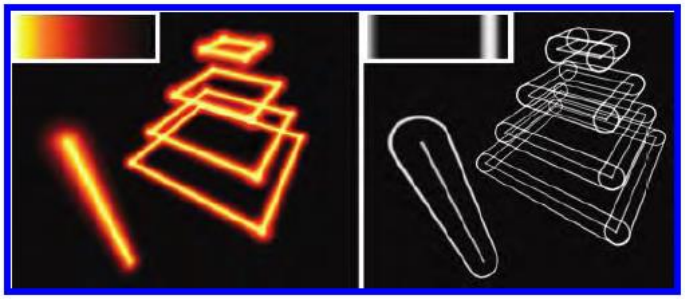
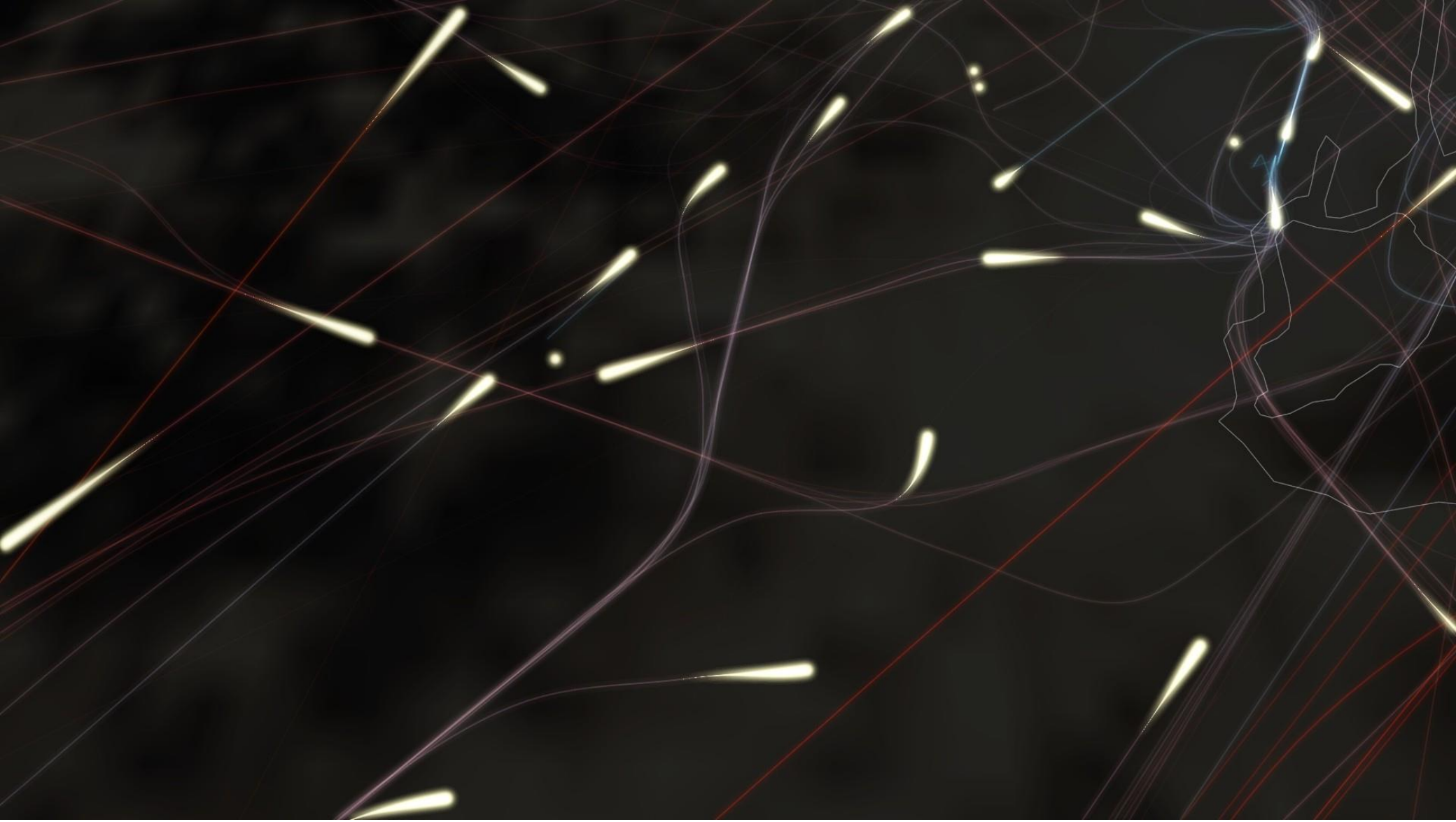


Figure 11.6. Volumetric lines rendered using a geometry shader–based geometry extrusion. Two different appearance gradients are shown here (small icon on the top-left of pictures).

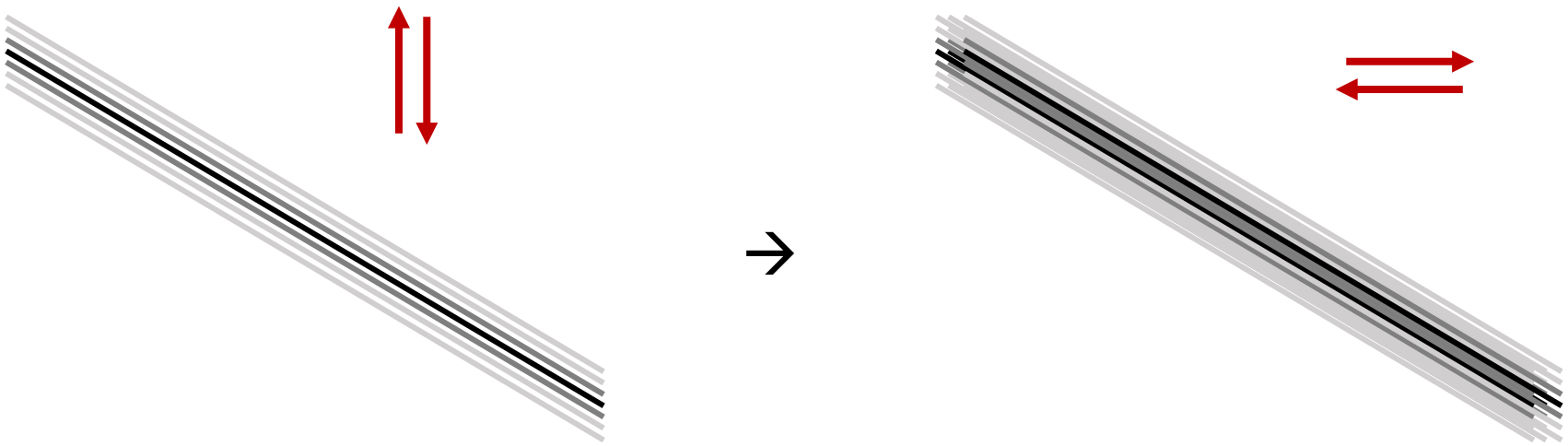
Patrick Cozzi and Christophe Riccio, <OpenGL Insights>

머리는 약간 복잡함

→ 분절된 선 하나하나마다 가상의 삼각형들을 4개씩 그린 후, 함수를 이용하여 면을 색칠함



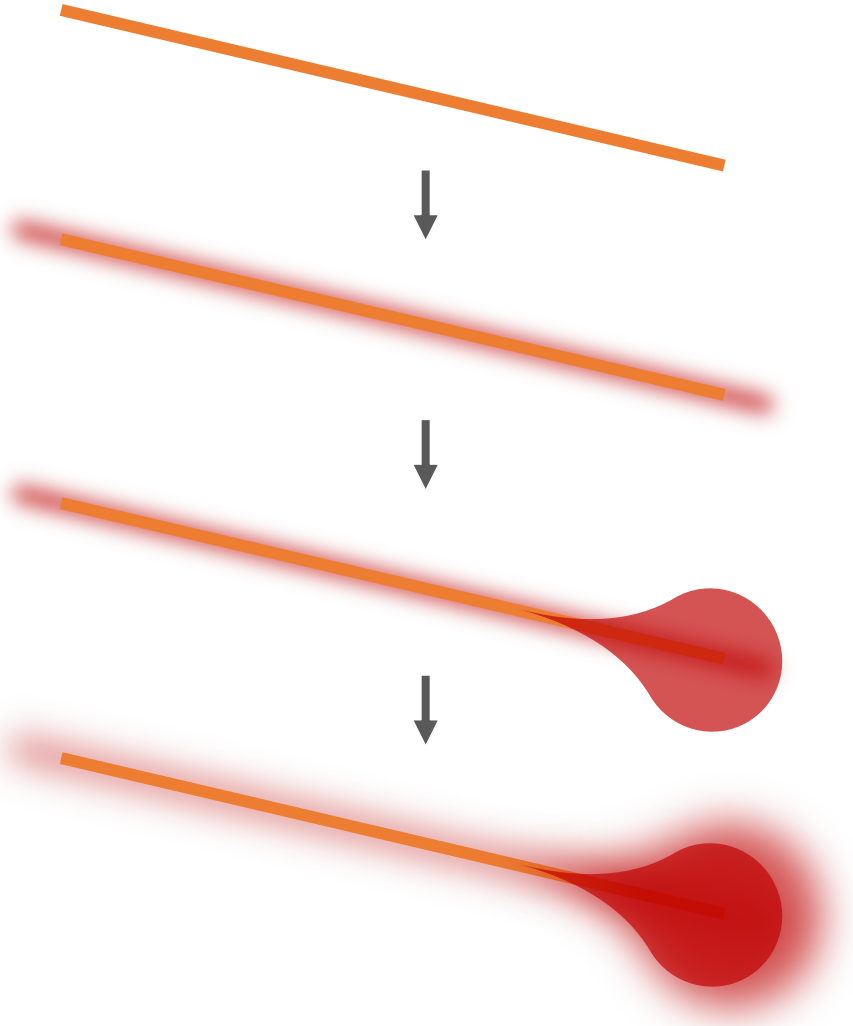
블러링



블러링

개체를 모두 그린 후 → 이미지 픽셀을 상하로 흔들고 → 다시 그 결과를 좌우로 흔든다

머리 길이 조정



선 그리기

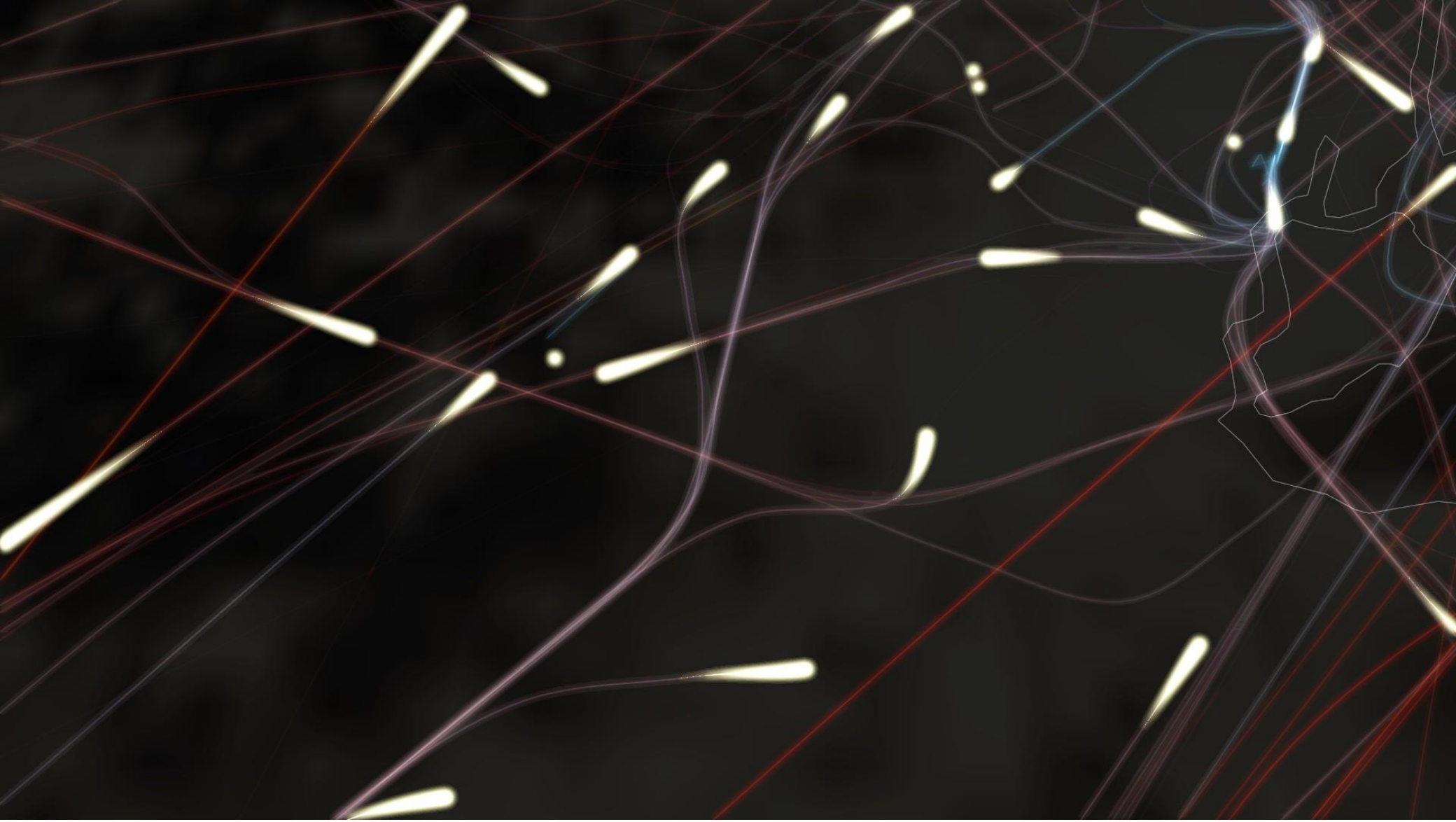
1차 블러링

머리 그리기

2차 블러링

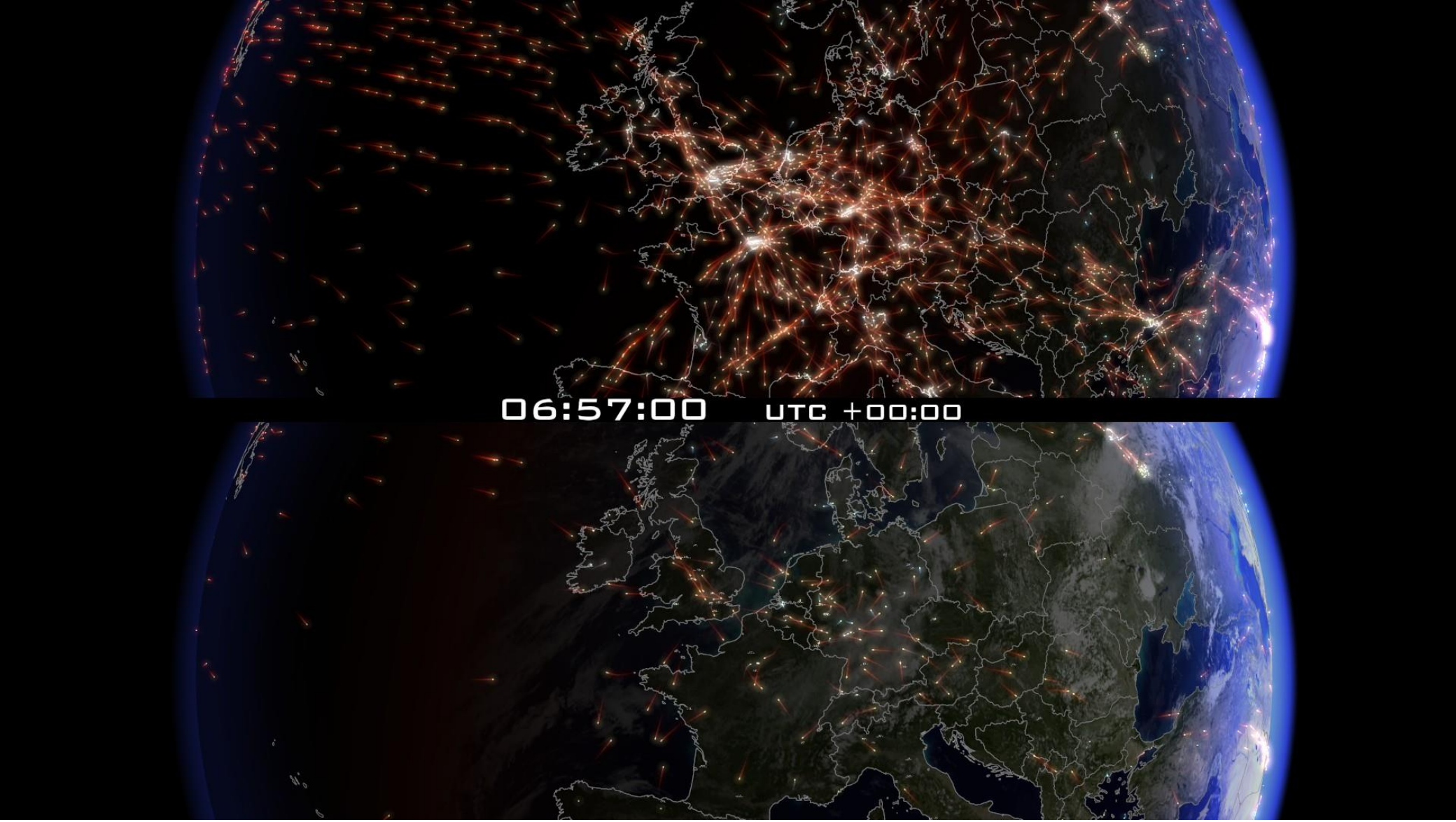
몸통(선) x a + 머리(면) x b + 블러링 x c

a, b, c 와 같은 혼합 계수를 조절해가면서 의도한대로 표현되는지 반복 확인



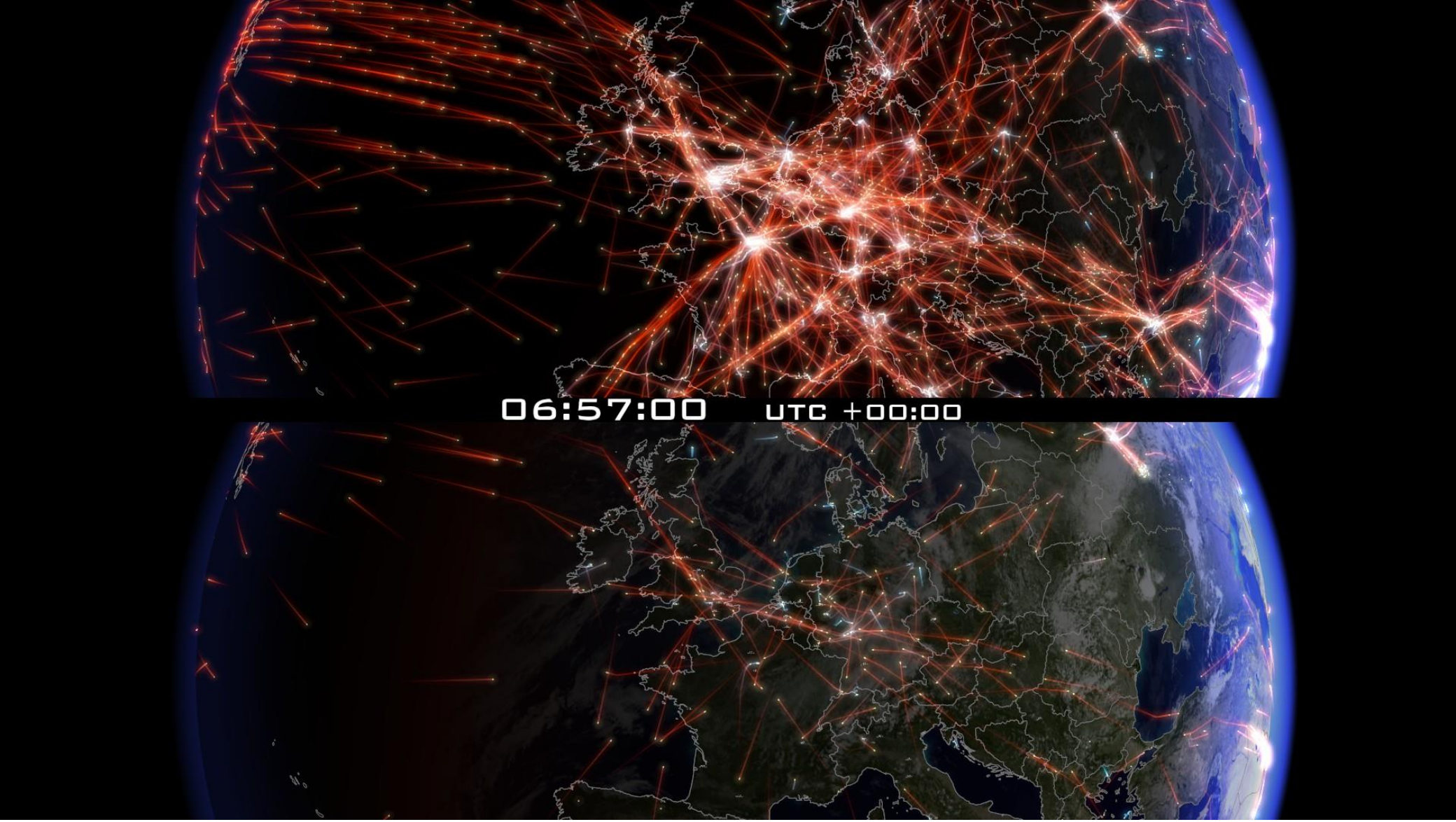
몸통(선) x a + 머리(면) x b + 블러링 x c

a, b, c 와 같은 혼합 계수를 조절해가면서 의도한대로 표현되는지 반복 확인



꼬리가 짧으면,

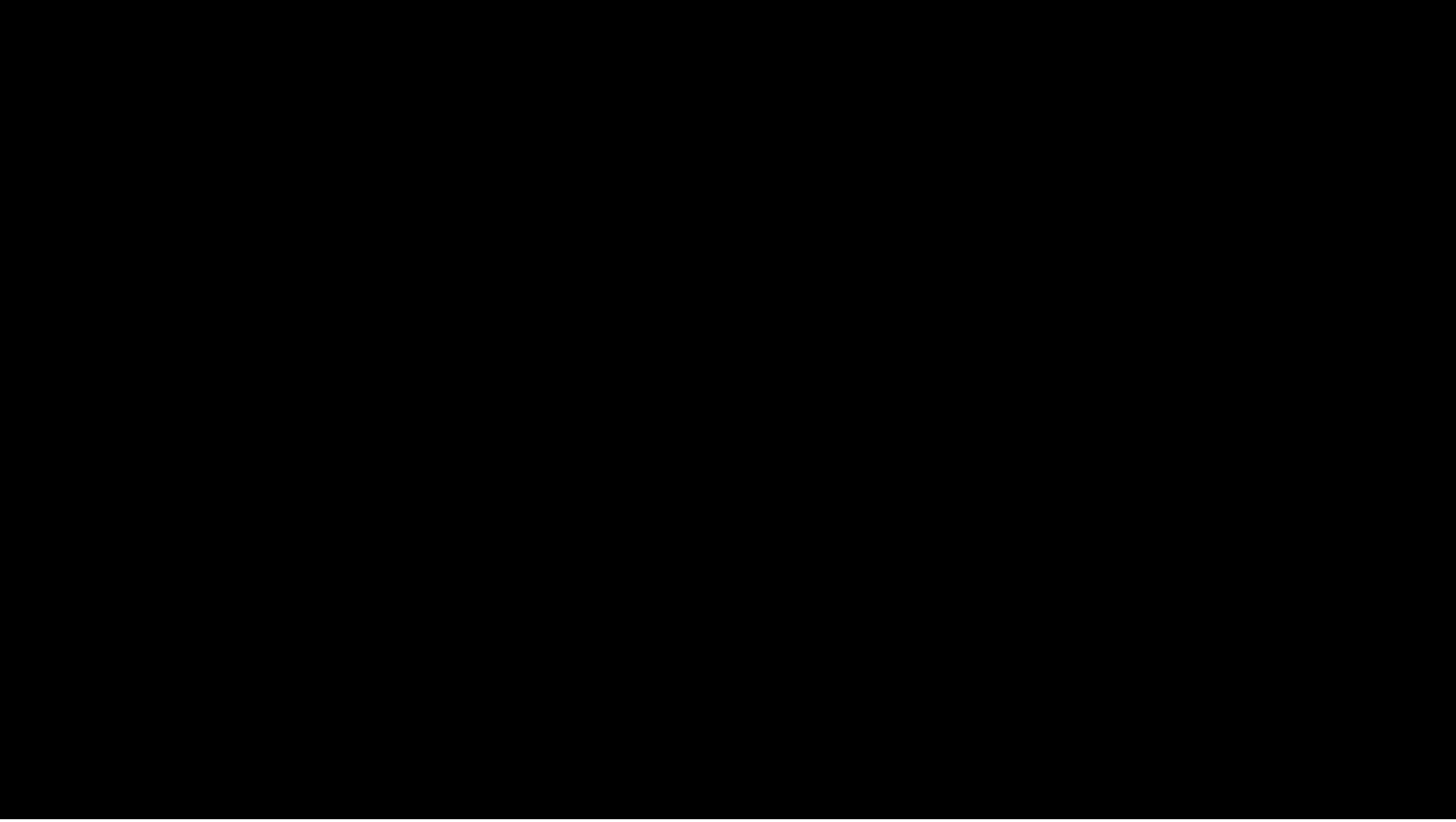
꼬리가 길면,



무수히 겹치는 선들의 색상 블렌딩은 최종 색상에도 영향을 끼침

→ 예측이 어려우므로 시행착오를 통해 꼬리길이와 색상을 동시 조정

항공기 운행량 감소 후 이착륙 경로가 단순하게 변함



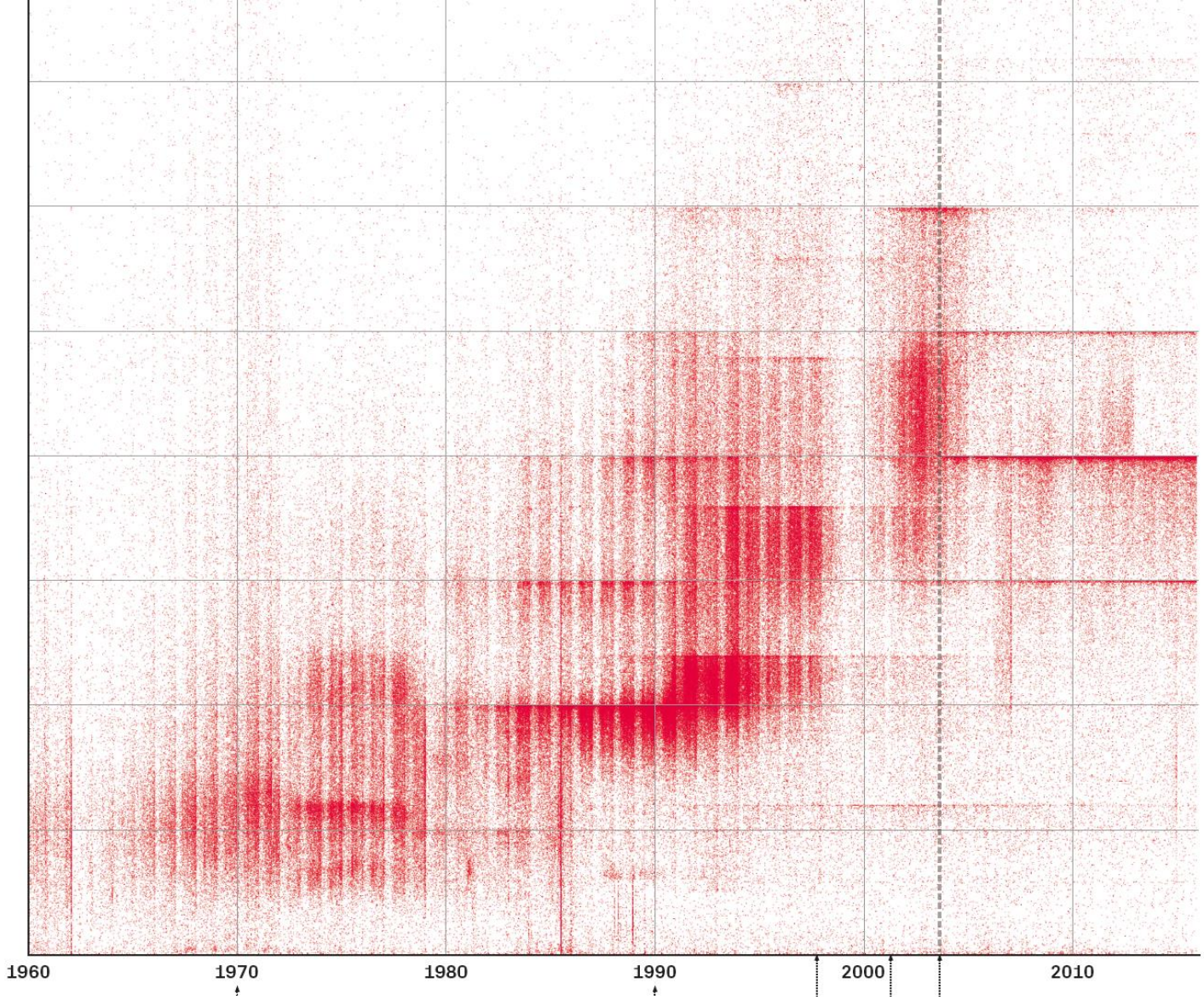
결과 영상

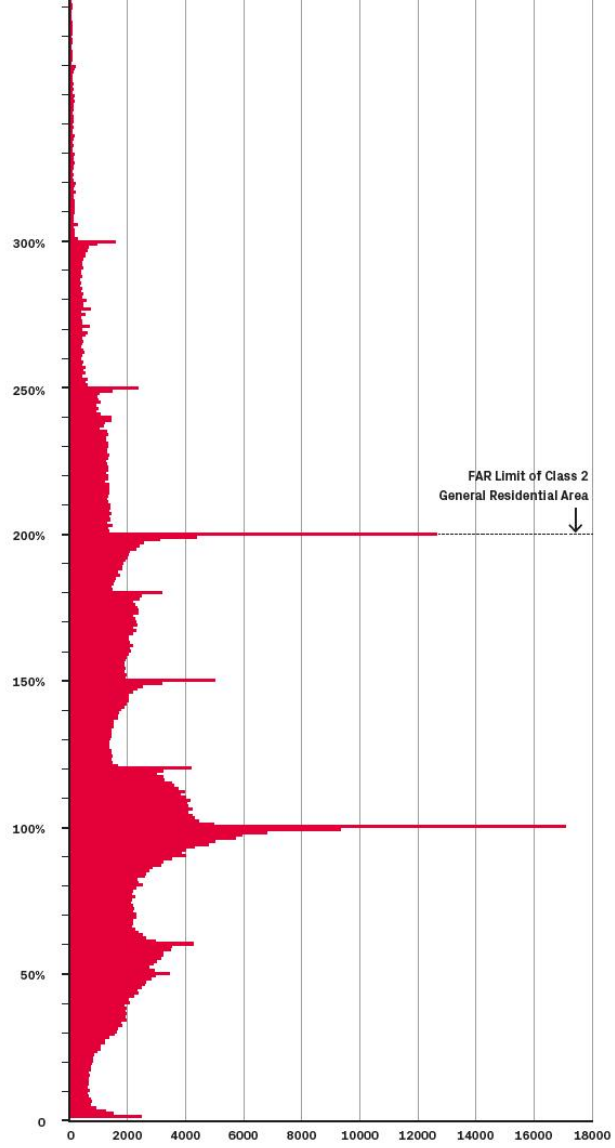
이상과 현실

빅데이터 시대의 표현 방식

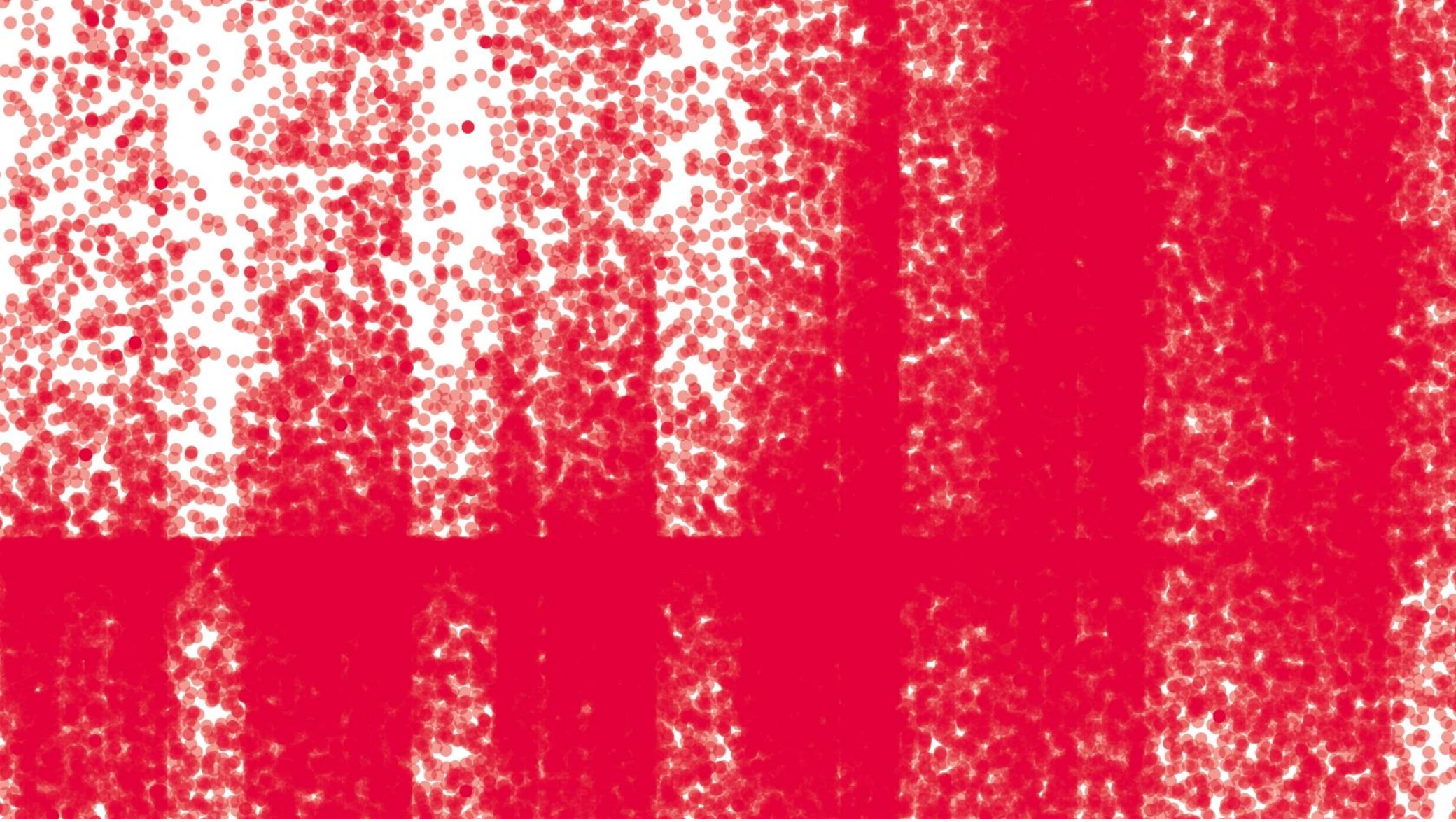
VS

클라이언트의 요구

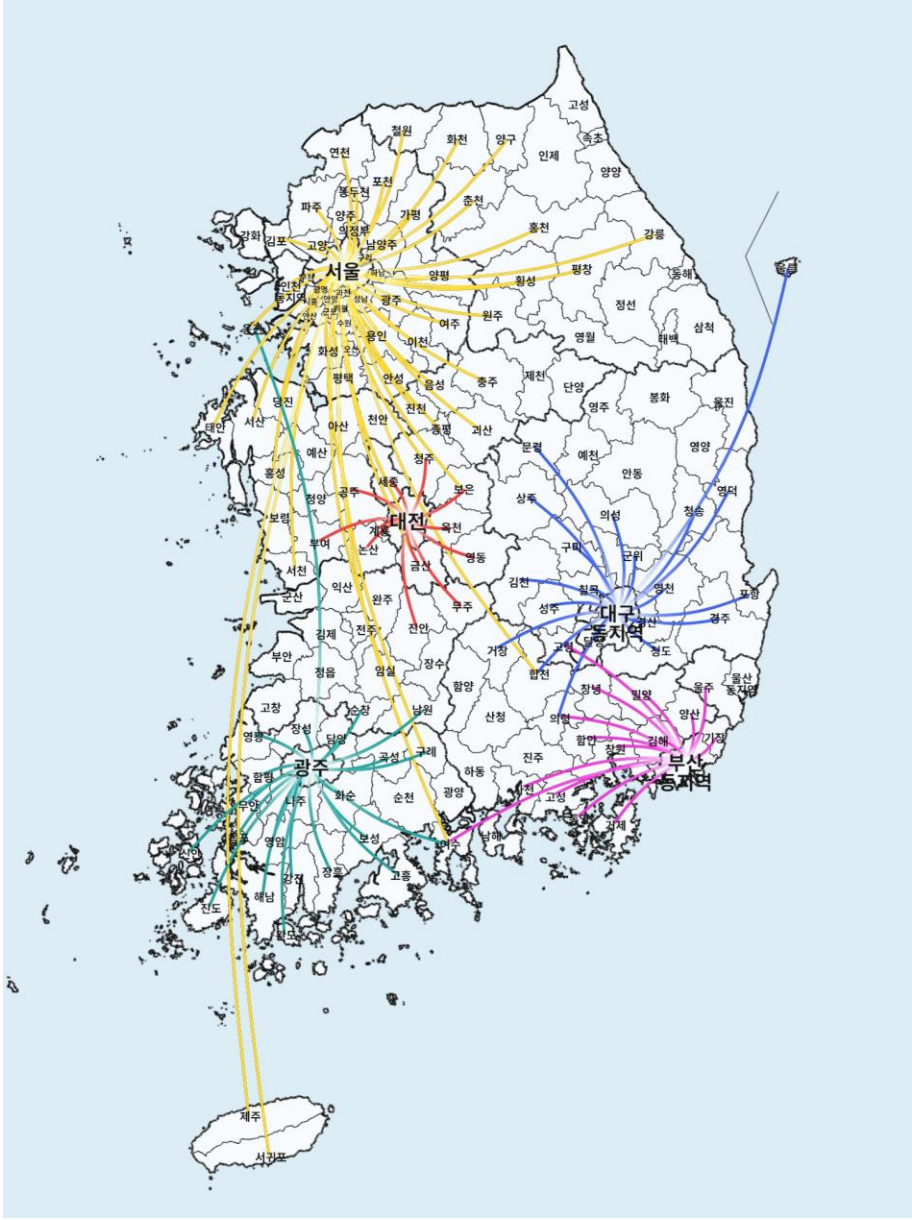




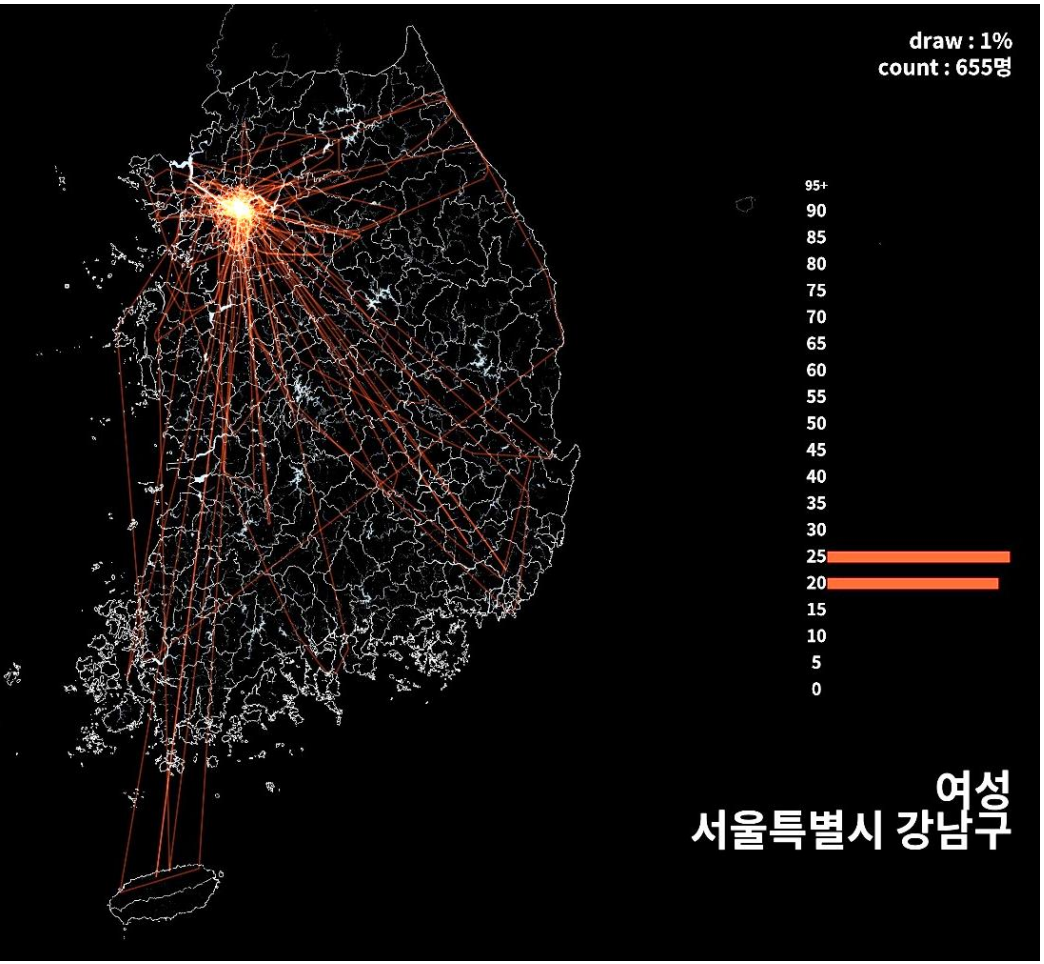
빅데이터의 시대 → 평균으로 묶지 않고 하나하나의 개별성을 표현

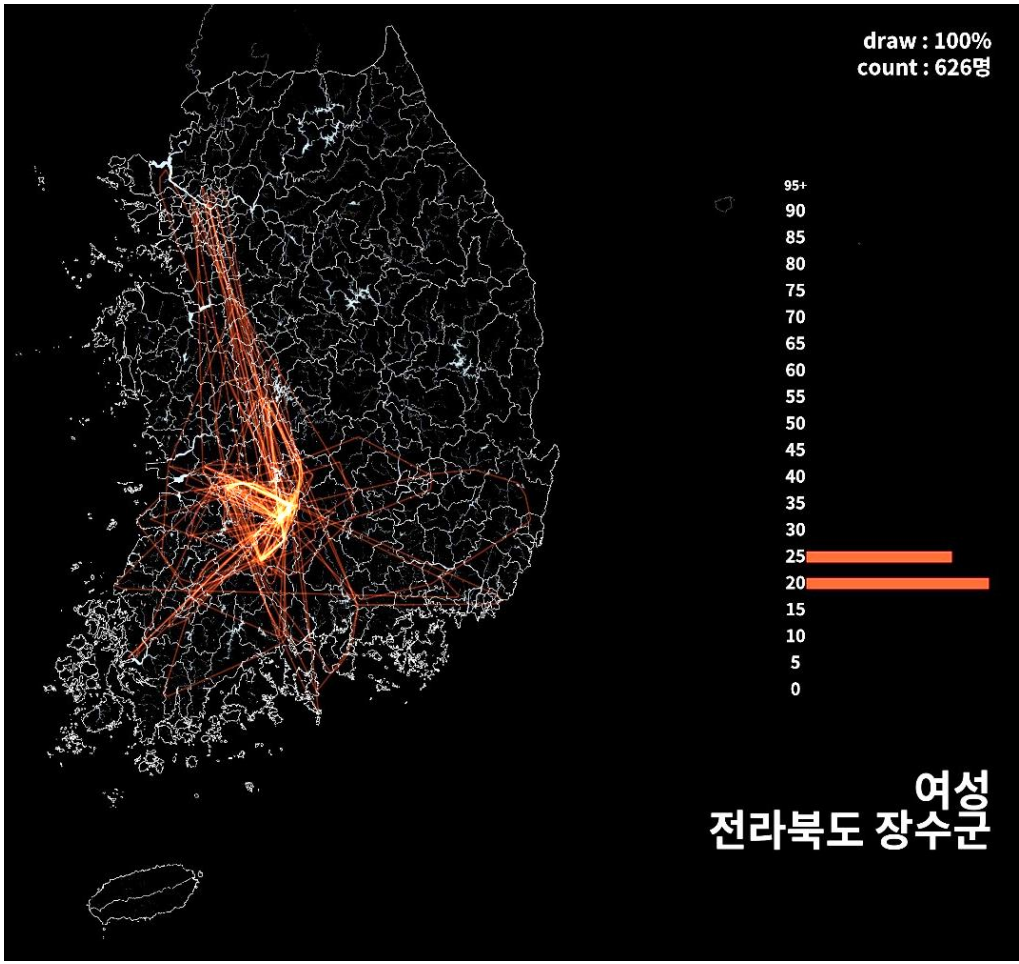


개별 요소들 각각의 특징이 살아있으면서도, 모여서 전체를 표현



“불필요한 정보들이 많다” → 명료한 커뮤니케이션이 어려움. 실제 인쇄되는 크기에 적합하지 않음





“단순한 숫자로 표현되지 않았다” → 짧은 시간에 정보를 습득하기 어려움

개별 요소의 자세한 표현

vs

데이터 시각화의 목적은 정보 전달


VWL