Journal of Environmental Science International
28(11); 1047~1059; November 2019

pISSN: 1225-4517 eISSN: 2287-3503
https://doi.org/10.5322/JESI.2019.28.11.1047

TECHNICAL NOTE

기상 및 대기질 정보의 3차원 표출 최적화를 위한 시제품 개발 연구

김건우[1,2] · 나하나[1] · 정우식[1]*

[1]인제대학교 대기환경정보공학과, [2]윈랩 주식회사

Prototype Development for Optimization Technique of 3D Visualization of Atmospheric Environmental Information

Gunwoo Kim[1,2], Hana Na[1], Woo-Sik Jung[1]*

[1]*Department of Atmospheric Environment Information Engineering, Inje University, Gimhae 50834, Korea*

[2]*Winlab Corporation, Busan 47881, Korea*

Abstract

To address the increase of weather hazards and the emergence of new types of such hazards, an optimization technique for three-dimensional (3D) representation of meteorological facts and atmospheric information was examined in this study as a novel method for weather analysis. The proposed system is termed as “meteorological and air quality information visualization engine” (MAIVE), and it can support several file formats and can implement high-resolution 3D terrain by employing a 30 m resolution digital elevation model. In this study, latest 3D representation techniques such as wind vector fields, contour maps, stream vector, stream line flow along the wind field and 3D volume rendering were applied. Implementation of the examples demonstrates that the results of numerical modeling are well reflected, and new representation techniques can facilitate the observation of meteorological factors and atmospheric information from different perspectives.

Key words : Meteorological Information, Air quality, 3D Visualization, Meteorological and air quality information visualization engine

1. 서 론

지구온난화에 따른 기상재해가 증가하고 새로운 유형의 재해들이 나타남에 따라(IPCC, 2007) 기상 분석을 위한 다양한 연구들이 활발히 진행되고 있으며 본 연구에서는 기상 분석을 위한 연구의 일환으로 기상 요소 및 대기질 정보를 3차원 적으로 표출해보고자 한다. 이와 관련한 선행 연구를 살펴본 결과, 국내에서는 국립기상과학원에서 연구 개발한 지구환경 3차원 가시화 시스템(지구 ON)이 있지만, 이는 프로젝터를 이용하여 구체 스크린에 투사하는 방식으로 우리가 연구하고자 하는 3차원 표출과는 거리가 멀었으며 기상청과 미래기후㈜에서 연구하고 개발한 3차원 기상표출 가시화 도구는 우리가 연구하고자 하는 방향과 비슷하지만 전 지구 규모의 기상

Received 5 November, 2019; **Revised** 14 November, 2019;
Accepted 18 November, 2019
***Corresponding author**: Woo-Sik Jung, Department of Atmospheric Environment Information Engineering, Inje University, Gimhae 50834, Korea
Phone : +82-55-320-3932
E-mail : wsjung1@inje.ac.kr

요소만 표출하고 있으며 표출 방식이 벡터와 등치선 정도로 그치는 아쉬움이 있었다(Im et al., 2010; Kim and Lee, 2014). 해외의 경우 국내와 비교하면 다양한 방식으로 연구를 하고 있었으며 그 중 도시 정보와 대기질 정보를 융합하거나(Roberto et al., 2011) 도시 정보와 허리케인 정보를 융합하여 표출하는 연구(Keqi et al., 2006) 등 도시 정보를 활용한 연구들이 많았다. 하지만 농도를 단순히 구체로 나타내거나 허리케인의 발달에 따른 수면의 증가만 다루는 등 표출 방식이 제한되어 있다는 아쉬운 점들도 발견할 수 있었다. 그 외에도 volume rendering을 통해 3차원 적으로 표출하는 연구의 경우(Fuchs and Hauser, 2009)는 다양한 요소를 나타내고 표출 수준도 뛰어나지만 기상 요소 가시화에 최적화되어 있지 않고 볼륨 렌더링에만 특화되어 있으며 프로그램이 복잡하다는 한계점이 있었다. 따라서 본 연구에서는 제반 대기 환경정보를 다루는 다수의 과학적 연구에서 생산된 여러 형태의 결과들을 효과적으로 표출하여 과학적 성과의 가시성과 시인성을 극대화할 수 있도록 기상 요소 및 대기질 정보를 2차원의 평면적 표출뿐만 아니라 3차원 공간에서의 분포 및 변화를 효과적으로 표출할 수 있도록 3차원 표출 최적화 기법이 적용된 시제품을 개발하는 연구를 수행하고자 하였다.

3차원 표출과 관련한 국내외의 논문들을 분석한 결과 전 지구 규모의 기상 요소만 표출하는 경우가 많았으며 표출 방식이 벡터와 등치선 정도로 그치는 아쉬움이 있었다. 따라서 본 연구에서 개발하고자 하는 기상 요소 및 대기질 정보 표출 프로그램(meteorological and air quality information visualization engine, MAIVE)에서는 고해상도 수치표고 모델 자료를 활용하여 3차원 지형을 구현하고 지형 위에서 기상 및 대기질 정보를 표출하고자 한다. 풍향, 풍속, 습도, 기온, 강수량 등과 같은 기상요소들과 미세먼지, 황사 농도와 같은 대기질 정보를 3차원 적으로 표출하며 파일 포맷은 NetCDF, Grib, Grib2, Ascii 등 대부분의 포맷을 지원한다. 그리고 상용 개발 엔진인 언리얼 엔진 4와 최신 렌더링 기법을 이용하여 비주얼적인 수준과 성능을 최적화하고 기존의 표출 방식뿐만 아니라 바람장을 따라 흐르는 벡터인 스트림 벡터와 유선인 스트림 라인을 추가하고, raymarching volume rendering 기법을 적용하여 구름 및 대기질 정보를 표출하고자 한다. 마지막으로 사례 분석을 통해 기상 요소와 대기질 정보의 3차원 표출의 최적화가 잘 이루어졌는지 살펴보았다.

2. 실험 방법

2.1. 자료

기상 요소 표출을 위한 사례를 위해 본 연구에선 WRF 수치 모델 결과를 이용하였다. Weather Research and Forecasting (WRF) model은 미국의 National Oceanic and Atmospheric Administration (NOAA)의 산하 기관인 National Centers for Environmental Prediction (NCEP)에서 개발한 기상 수치예보 모델이자 기상 시뮬레이션 시스템이다. 중규모 기상에 대한 분석과 예보를 향상하기 위해 새로운 예보 모델과 자료동화 시스템을 개발하였으며 이 과정에서 National Center for Atmospheric Research (NCAR)의 Mesoscale and Microscale Meteorology (MMM) division, Earth System Research Laboratory (ESRL), Air Force Weather Agency (AFWA) 등 많은 기관과 대학의 합동 노력으로 만들어졌다.

또한, 대기질 정보 표출 사례를 위해 본 연구에서는 Community Multiscale Air Quality (CMAQ) 모델 결과를 활용하였다. CMAQ이란 미국 환경보건청(United States Environmental Protection Agency, US EPA)이 기존의 대기오염 모델들의 문제점을 보완하여 제3세대 대기오염 모델로 제시한 모델로서 현재도 활발하게 개발이 진행 중인 모델이다. 전처리 모델과 화학수송 모델을 합친 모델로서 모듈구조로 되어 있어 상호 호환이 가능하고 모델링 영역을 국지 및 지역 규모로 다양하게 정할 수 있으며 화학반응, 이류, 확산 등에 최신 이론을 반영하고 황화합물, 오존, 에어로졸의 예측이 가능하다는 장점이 있다.

2.2. Unreal engine 4를 활용한 연구 및 개발

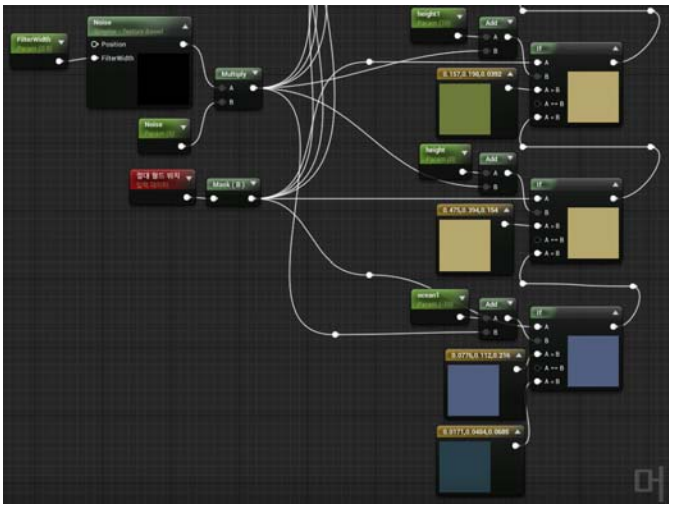
상용 엔진인 유니티 엔진과 언리얼 엔진 4를 비교 분석한 결과, 본 연구에서는 그래픽 표출의 측면에서 우수한 것으로 평가되는 언리얼 엔진 4를 활용하여 연구를 진행하기로 결정하였다(Table 1).

2.2.1. 3차원 표출을 위한 지형 표현

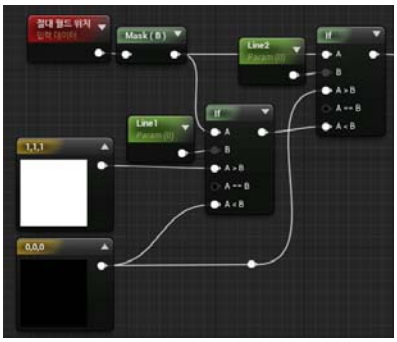
기상 요소 및 대기질 정보의 3차원 표출을 위해 가장

Table 1. Pros and cons of unity engine and unreal engine

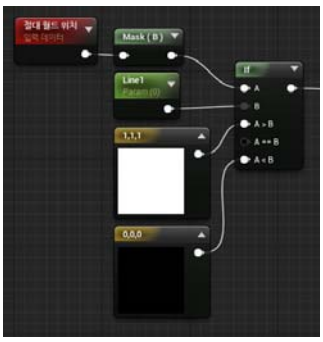
Tools	Unity engine	Unreal engine
Pros	• Multi platform build • Easy UI for development • Low requirements • Asset store • License free if non-commercial	• Multi platform build • Blueprint • Partnership with MS visual studio • License free if non-commercial • Fast application of the latest technology
Cons	• Advanced techniques limited • Multithread limited • Weak security	• High requirements • Blueprint learning time spent • Few asset



(a) Terrain skin color by height



(b) Terrain contour skin



(c) Terrain mask skin

Fig. 1. Blueprint of terrain skin by material function.

먼저 3차원 지형을 구현할 필요성이 있다. 본 연구에서는 30 m 해상도의 수치표고 모델(Digital Elevation Model, DEM) 자료를 활용하여 사례별 지형을 표출하였다. DEM 자료를 활용하기 위해 원하는 영역의 DEM 자료를 높이맵 형식으로 추출할 필요가 있으며 본 연구에서는 global mapper 프로그램을 이용하여 30 m 해상도의

```
ifstream file_windU("D:\\Data\\TD\\U.csv");
ifstream file_windV("D:\\Data\\TD\\V.csv");
char cstr[2048];
TArray<FString> Array;
TArray<FString> Array2;
WindAngle.Empty();
WindSpeed.Empty();
float angle, speed = 0.0;
for (int d = 0; d < 15; ++d)
{
  for (int z = 0; z < 29; ++z)
  {
    for (int y = 0; y < 183; ++y)
    {
      file_windU.getline(cstr, 2048);
      FString t(cstr);
      t.ParseIntoArray(Array, TEXT(","), true);
      file_windV.getline(cstr, 2048);
      FString t2(cstr);
      t2.ParseIntoArray(Array2, TEXT(","), true);
      for (int x = 0; x < 183; ++x)
      {
        angle = UKismetMathLibrary::Atan2(FCString::Atof(*(Array[x])),
            FCString::Atof(*(Array2[x])));
        speed = sqrt(FCString::Atof(*(Array2[x])) * FCString::Atof(*(Array2[x])) +
            FCString::Atof(*(Array[x])) * FCString::Atof(*(Array[x])));
        WindAngle.Emplace(angle * 180 / PI);
        WindSpeed.Emplace(speed);
      }
    }
    file_windV.getline(cstr, 2048);
    file_windV.getline(cstr, 2048);
    file_windU.getline(cstr, 2048);
  }
}
```

Fig. 2. Source code to read wind uv from file.

DEM 자료를 추출하였다. 추출한 DEM 높이맵 자료를 언리얼 엔진 4의 landscape 기능을 통해 적용하여 3차원 지형을 생성하였다. 추가로 언리얼 엔진의 material 기능을 활용하여 지형의 높이에 따른 색을 입히는 기본 스킨과 바다와 육지를 구별하는 마스크 스킨, 바다와 육지의 경계선을 표시하는 윤곽선 스킨을 적용하였다(Fig. 1).

2.2.2. 기상 요소 표현

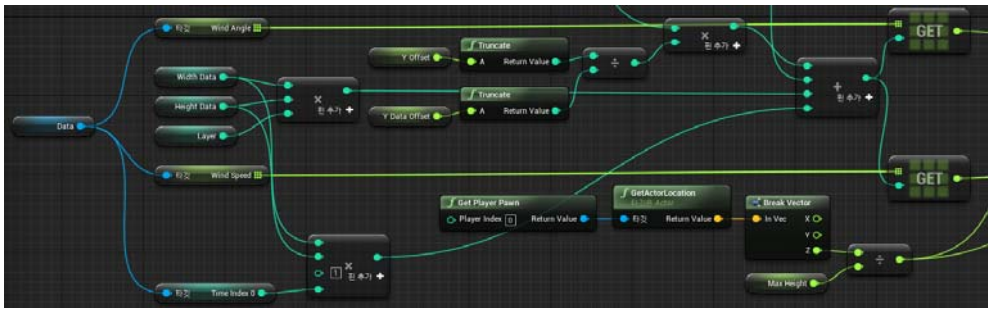
2.2.2.1. 바람 벡터장

바람을 나타내는 가장 기본적인 바람장을 표출하기 위해 C++ 소스 코드를 활용해 raw 데이터로부터 바람의 u, v 성분을 추출하여 읽어오게 하였다(Fig. 2). 읽어온 u, v 값을 이용하여 풍향과 풍속을 계산하여 배열에 저장하고 이를 언리얼 엔진에서 읽어 들여 바람장 각각의 벡터에 반영하도록 프로그램 하였다(Fig. 3).

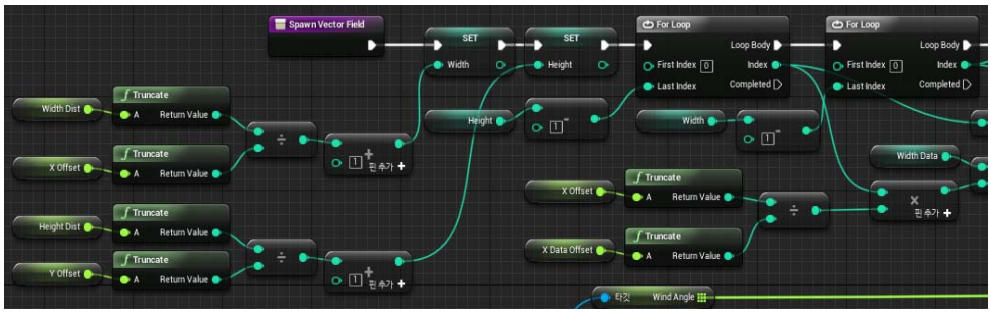
추가로 바람 벡터장을 3차원 지형상에서 표출하기 위해 다양한 기법을 시도하였다. 풍속에 따른 벡터 색과 단일 벡터 색을 구현하였으며 벡터장의 격자 간격과 벡터의 크기를 동적으로 변경할 수 있도록 하였다.

2.2.2.2. 스트림 벡터 & 스트림 라인

바람 벡터장의 경우 바람의 전체적인 경향을 파악하기에는 좋으나 바람의 흐름을 파악하기에는 다소 아쉬운 부분이 나타났다. 따라서 바람장을 따라 흐르는 바람

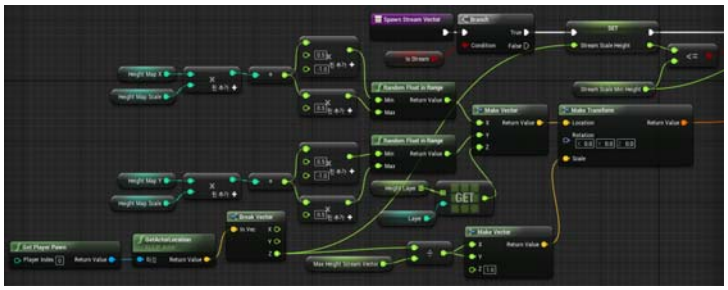


(a) Read a value from data in blueprint

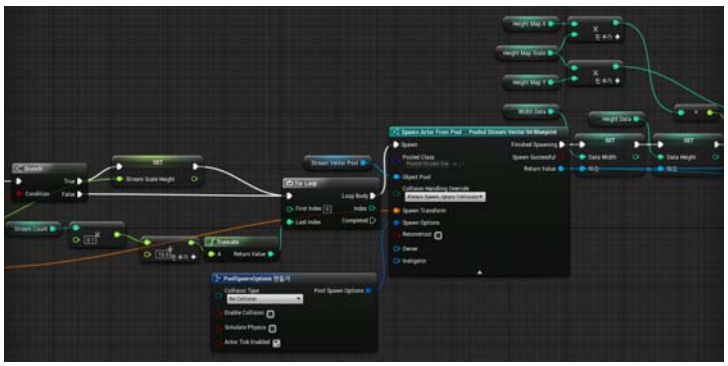


(b) Create wind vector in blueprint

Fig. 3. Read and apply a value from data in blueprint.

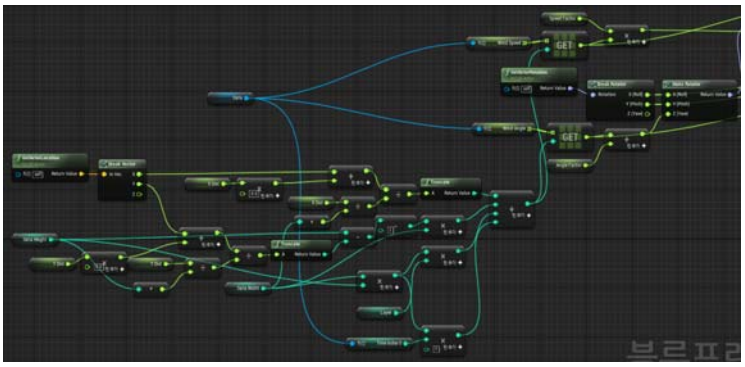


(a) Specify location of stream vector in blueprint

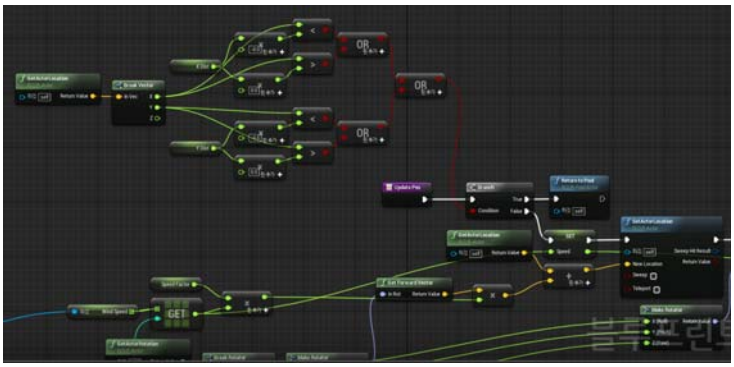


(b) Create stream vector in blueprint

Fig. 4. Spawn stream vector in blueprint.



(a) Calculate current position in blueprint



(b) Set location of stream vector in blueprint

Fig. 5. Update location stream vector in blueprint.

벡터인 스트림 벡터와 바람장을 따라 흐르는 선인 스트림 라인을 표출하였으며 이는 고정된 바람 벡터장과 달리 이동하면서 변화하므로 바람의 흐름을 파악하는 데 큰 도움이 된다. 각각의 스트림 벡터와 스트림 라인은 불특정 위치에서 생성되어(Fig. 4) 그 위치의 풍향, 풍속을 반영하여 움직이게 된다(Fig. 5). 스트림 벡터의 경우 바람 벡터장과 마찬가지로 풍속에 따른 벡터 색과 단일 색을 선택할 수 있도록 하였으며 스트림 라인은 단일 색만 표출하였다.

2.2.2.3. Contour map

Contour map은 기상 요소 및 대기질 정보와 같은 다양한 데이터 값을 색으로 표현한 지도로서 데이터의 경향과 분포를 시각적으로 쉽게 볼 수 있게 하는 장치이다. Fig. 6의 소스 코드는 기상 요소 중 하나인 풍속을 contour 텍스쳐로 만드는 코드이며 데이터의 범위를 최소값인 MinValue와 최대값인 MaxValue의 차로 계산한 뒤 풍속 데이터를 0.0부터 1.0까지 정규화 하여 범례 색깔의 색과 대응시킨다. 대응된 색 정보는 Contour-Temp 배열에 저장한 뒤 UpdateTextureRegions 함수를 이용해 배열의 값을 텍스쳐로 생성한다.

2.2.2.4. 볼륨 렌더링

본 연구에서는 구름 및 미세먼지 등 3차원 적으로 표출하기 어려운 변수들을 volume rendering을 이용해 표출하고자 하였다. volume rendering은 3차원 데이터를 2차원으로 화면 투영하여 보여주기 위한 기술의 집합이라 할 수 있다. volume rendering을 위한 다양한 기법들이 존재하는데 여기서는 최신 기술인 raymarching 알고리즘을 적용하기로 하였다.

Raymarching 알고리즘의 기본 개념은 광선을 볼륨에 비췄을 때 볼륨을 통과하는 광선을 평가하는 것으로 볼륨과 광선이 교차하는 각 픽셀에 대해 불투명도와 색상을 반환하는 것을 의미한다(Ryan, 2016). 이 과정은

```
ContourTemp.Init(FColor(0, 0, 0, 255), 183 * 183);
if (ContourIndex == 0) {
   float range = MaxValue - MinValue;
   int value = 0;
   int index2 = 0;
   int index3 = 0;
   const FColor* FormatedImageData = static_cast<const FColor*>(ColorBar->PlatformData->
            Mips[0].BulkData.LockReadOnly());
   for (int y = 0; y < 183; ++y)
   {
      for (int x = 0; x < 183; ++x)
      {
         index3 = x + (183 - y - 1) * 183 + (183 * 183 * HeightIndex) +
               (183 * 183 * 1 * TimeIndex);
         index2 = x + y * 183;
         value = (int)(((WindSpeed[index3] - MinValue) / range) * 1648);
         ContourTemp[index2] = FormatedImageData[value];
      }
   }
   UpdateTextureRegions(ContourTexture, (int32)0, (uint32)1, ContourTextureRegion, (uint32)(4 * 183), (uint32)4,
(uint8*)ContourTemp.GetData(), false);
   ColorBar->PlatformData->Mips[0].BulkData.Unlock();
}
```

Fig. 6. Create contour texture in source code.

```
float numFrames = XYFrames * XYFrames;
float accumdist = 0;
float3 localcamvec = normalize(mul(Parameters.CameraVector, Primitive.WorldToLocal));
float StepSize = 1 / MaxSteps;
for (int i = 0; i < MaxSteps; i++)
{
   float cursample = PseudoVolumeTexture(Tex, TexSampler, saturate(CurPos), XYFrames, numFrames).r;
   accumdist += cursample * StepSize;
   CurPos += -localcamvec * StepSize;
}
return accumdist;
```

(a) Calculate transmittance

```
float numFrames = XYFrames * XYFrames;
float curdensity = 0;
float transmittance = 1;
float3 localcamvec = normalize(mul(Parameters.CameraVector, Primitive.WorldToLocal)) * StepSize;
float shadowstepsize = 1 / ShadowSteps;
LightVector *= shadowstepsize;
ShadowDensity *= shadowstepsize;
Density *= StepSize;
float3 lightenergy = 0;
for (int i = 0; i < MaxSteps; i++)
{
   float cursample = PseudoVolumeTexture(Tex, TexSampler, saturate(CurPos), XYFrames, numFrames).r;
   if( cursample > 0.001)
```

```
    {
        float3 lpos = CurPos;
        float shadowdist = 0;
        for (int s = 0; s < ShadowSteps; s++)
        {
            lpos += LightVector;
            float lsample = PseudoVolumeTexture(Tex, TexSampler, saturate(lpos), XYFrames,
                                numFrames).r;
            shadowdist += lsample;
        }
        curdensity = saturate(cursample * Density);
        float shadowterm = exp(-shadowdist * ShadowDensity);
        float3 absorbedlight = shadowterm * curdensity;
        lightenergy += absorbedlight * transmittance;
        transmittance *= 1-curdensity;
    }
    CurPos -= localcamvec;
}
return float4( lightenergy, transmittance);
```

(b) Adding calculate shadow step

Fig. 7. Volume rendering by shader code.

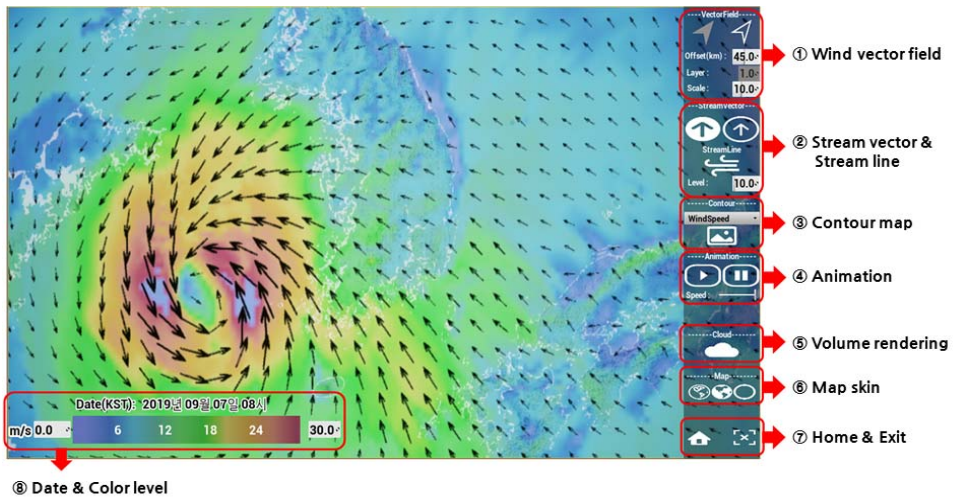


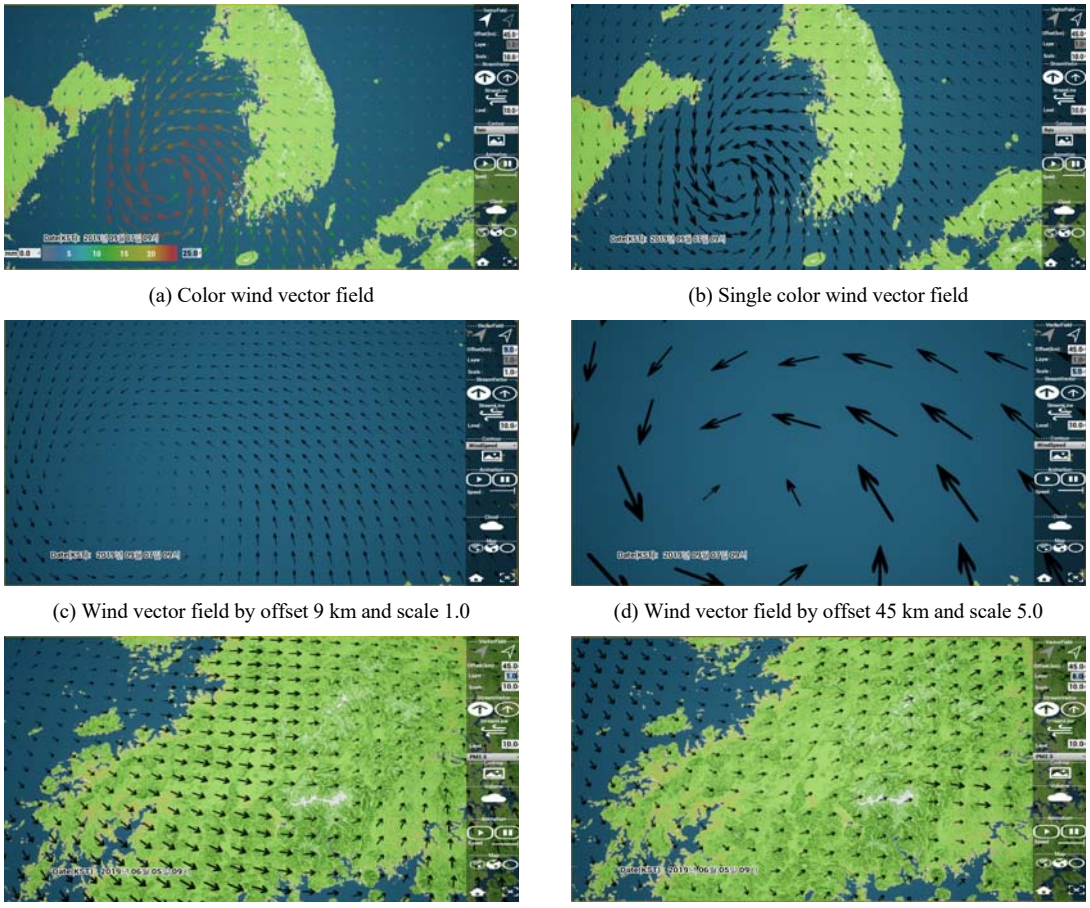
Fig. 8. Menu composition of MAIVE.

크게 광 흡수에 의한 불투명도 생성과정과 조명, 산란에 의한 색상 생성과정으로 이루어져 있으며 이를 shader를 이용하여 표출하였다(Fig. 7).

3. 결 과

Fig. 8은 MAIVE 프로그램의 메뉴 구성을 나타내고 있다.

① 바람 벡터장을 제어하는 메뉴이다. 좌측의 화살표 버튼은 풍속에 따른 색을 입힌 바람 벡터장을 화면상에 켜고 끄는 역할을 하며(Fig. 9의 a) 우측의 화살표는 검은색의 단일 바람 벡터장을 표출하고 지우는 역할을 한다(Fig. 9의 b). offset의 경우 벡터장의 간격을 설정할 수 있으며 이 크기는 raw 데이터의 격자 간격보다 낮을



(a) Color wind vector field

(b) Single color wind vector field

(c) Wind vector field by offset 9 km and scale 1.0

(d) Wind vector field by offset 45 km and scale 5.0

(e) Wind vector field by layer 1

(f) Wind vector field by layer 8

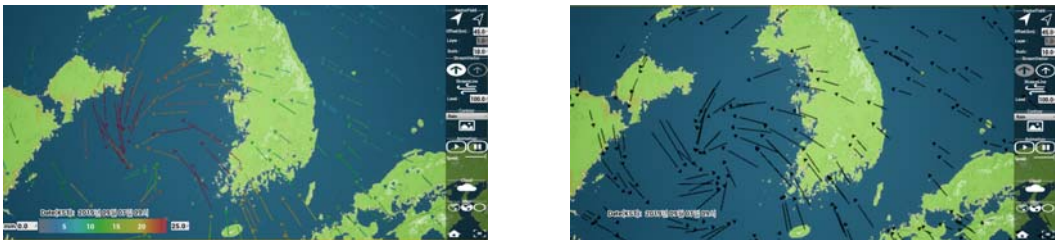
Fig. 9. Wind vector field function description.

수는 없다. 또한, scale은 바람 벡터장의 크기를 조절하며 숫자가 클수록 전반적인 벡터의 크기가 커진다(Fig. 9의 c, d).

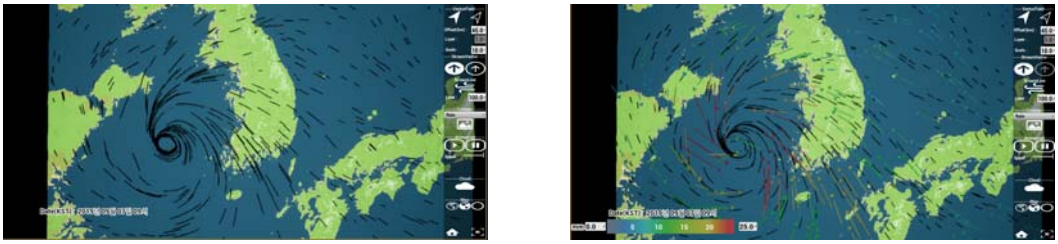
② 스트림 벡터와 라인을 제어하는 메뉴이다. 좌측의 흰색 배경의 검은 화살표 버튼은 바람 벡터장과 같게 풍속에 따른 색의 스트림 벡터를 화면상에 켜고 끄는 역할을 하며(Fig. 10의 a), 우측의 검은 배경의 흰색 화살표는 검은색으로 통일한 스트림 벡터를 표출하는 역할을 한다(Fig. 10의 b). 스트림 라인 버튼은 스트림 라인을 화면상에 표출하고 지우는 역할을 하며(Fig. 10의 c) 스트림 벡터와 스트림 라인은 동시에 표출될 수도 있다(Fig. 10의 d). level의 경우 현재 화면상에 생성되는 스트림 벡터와 라인의 개수를 제어하는 메뉴로써 level이 높을수록 많은 양의 스트림 벡터와 라인이 표출되지만 그만큼 부하가 많이 걸리기 때문에 사용자의 실행 환경에 맞춰서 조절할 필요성이 있다(Fig. 10의 e, f).

③ Contour map을 제어하는 메뉴로써 버튼을 선택하면 지형에 contour map이 융합되어 표출하게 된다. 드롭 메뉴에는 현재 보고자 하는 기상 요소 및 대기질 정보를 선택할 수 있도록 하였다(Fig. 11). 또한, contour map은 모든 지형 스킨에서 적용할 수 있어 사용자가 원하는 그림에 따라 스킨을 활용할 수 있도록 하였다.

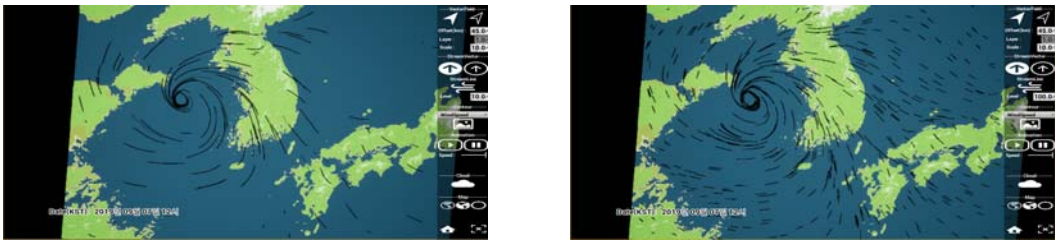
④ 시간에 따른 애니메이션을 제어하는 메뉴로써 좌측 실행 버튼을 선택하면 시간이 지나면서 현재 나타나고 있는 기상 요소 및 대기질 정보가 그 시간에 맞도록 변화하게 된다. 우측 정지 버튼을 선택하면 그 시간에 멈추



(a) Color stream vector (b) Single color stream vector

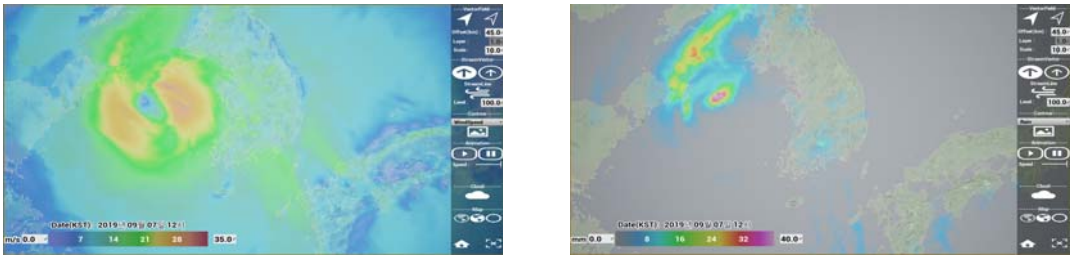


(c) Stream line (d) Stream line and stream vector

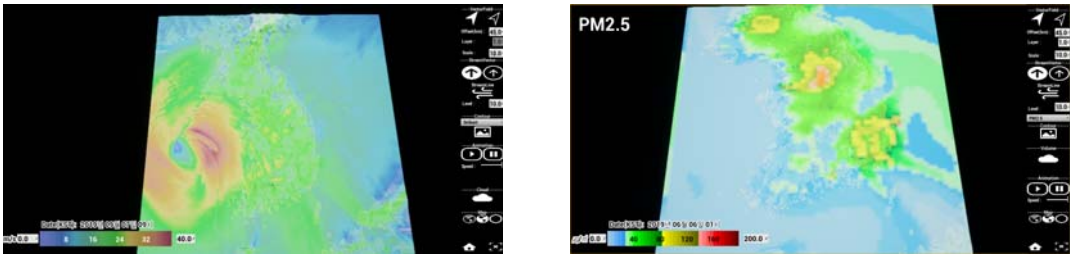


(e) Stream line by level 10 (f) Stream line by level 100

Fig. 10. Stream vector and stream line function description.

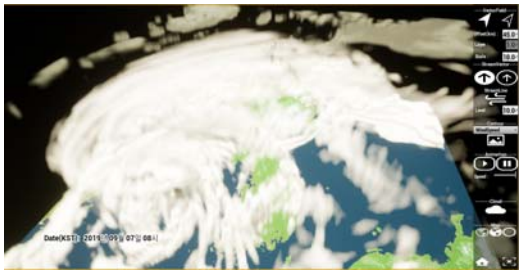


(a) Wind speed contour map (b) Rain contour map



(c) 3-second gust contour map (d) $PM_{2.5}$ contour map

Fig. 11. Contour map function description.



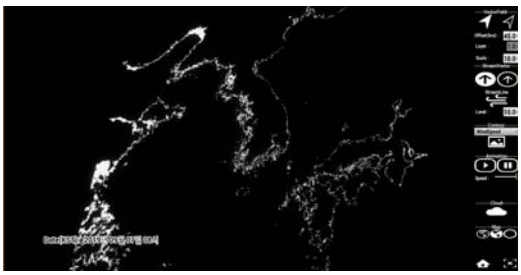
(a) Cloud volume rendering

(b) SO_2 Volume rendering

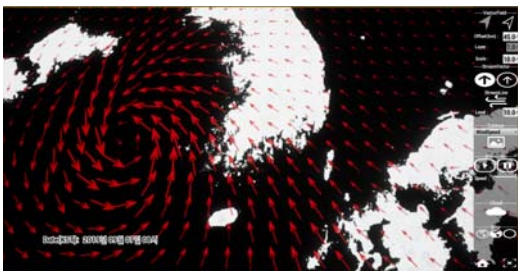
Fig. 12. Volume rendering function description.

(a) Base skin

(b) Mask skin

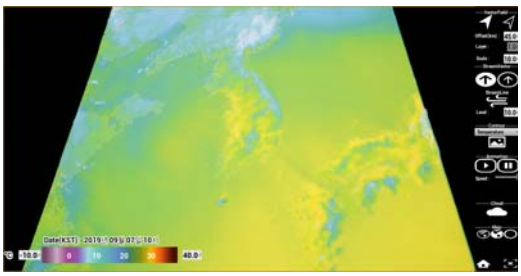


(c) Line skin

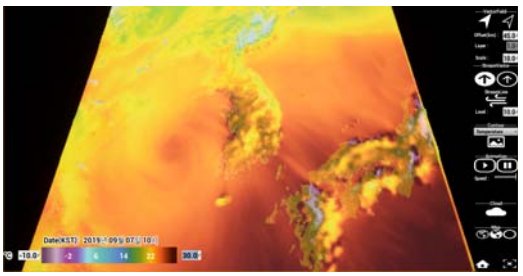


(d) Wind vector field on mask skin

Fig. 13. Map skin function description.



(a) Temperature contour map by range -10℃ ~ 40℃



(b) Temperature contour map by range -10℃ ~ 30℃

Fig. 14. Color legend function description.

게 되며 아래의 speed 바를 통해 애니메이션의 속도를 조절할 수 있도록 하였다.

⑤ 볼륨 렌더링을 제어하는 메뉴로써 버튼을 선택할 시 구름 및 미세먼지 등 원하는 요소의 볼륨 렌더링을 표출하고 지울 수 있도록 하였다(Fig. 12).

⑥ 지형의 스킨을 변경하는 메뉴로 첫 번째 버튼은 높이에 따른 색을 입힌 기본 스킨, 두 번째 버튼은 육지는 흰색, 바다는 검은색으로 표현하는 마스크 스킨, 세 번째 버튼은 육지와 바다의 경계를 흰색으로 표시하는 윤곽선 스킨으로 전환할 수 있다(Fig. 13).

⑦ 좌측 버튼은 사례를 고를 수 있는 홈 화면으로 돌아가는 임무를 수행하며 우측 버튼은 프로그램을 종료하는 임무를 수행한다.

⑧ 데이터상의 시간을 나타내고 색깔 범례를 제어하는 메뉴이다. 시간은 애니메이션을 실행시킬 경우 변화하게 된다. 색깔 범례의 경우 풍속, 기온, 강수량 등 여러 요소에 따라 다른 범례를 사용하고 있으며 좌측과 우측의 최소, 최대값 입력을 통해 색깔 범례의 범위를 지정할 수 있게 하여 사용자가 원하는 구간을 강조해서 볼 수 있도록 하였다(Fig. 14).

4. 결 론

본 연구에서는 지구온난화로 인한 기상재해의 증가에 따른 기상 분석의 하나의 방법으로 기상 요소 및 대기질 정보의 3차원 표출의 최적화 기법을 연구하고 기법을 적용한 MAIVE를 개발하였다.

상용 엔진 중 업데이트가 활발히 이루어지고 무료 라이선스이며 비주얼적인 면에서 뛰어난 unreal engine 4를 사용하여 기본적인 3차원 지형을 위해 DEM 자료를 활용하여 입체 지형을 표출하였으며 높이에 따른 색을 달리하는 기본 스킨과 육지와 바다의 색을 달리한 마스크 스킨, 육지와 바다의 경계선을 표시하는 윤곽선 스킨을 적용하였다. 기상 요소 및 대기질 정보를 3차원 표출하기 위해 다양한 기법들을 연구하였으며 바람과 같은 벡터값은 기본적인 벡터장뿐만 아니라 바람을 따라 흐르는 벡터인 스트림 벡터와 유선인 스트림 라인 기법을 추가하여 바람의 흐름을 보다 직관적으로 볼 수 있도록 하였다. 기온, 강수량 등의 2차원 적인 데이터 값은 contour map을 통해 그 분포와 경향을 파악할 수 있도록 하였으며 지형의 다양한 스킨과 융합하여 사용자가 원하는 그림을 선택하도록 하였다. 구름, 대기질 정보와 같은 3차원 데이터값은 최신 기법인 raymarching 알고리즘을 적용한 볼륨 렌더링 기법으로 3차원 상에 표출할 수 있었다.

사례의 표출을 확인한 결과 수치 모델 결과를 잘 반영하는 것을 알 수 있었고, 새로운 표출 기법을 통해 다른 관점에서 기상 요소 및 대기질 정보를 분석할 수 있다는 것을 확인하였다. 또한, 본 연구에서는 WRF 모델과 CMAQ 모델 결과를 이용하여 한정된 사례만 분석하였으나 다른 수치 모델 결과나 데이터가 주어진다면 더욱 다양한 사례들을 분석해볼 수 있을 것으로 판단된다.

마지막으로 본 연구를 통해 기상 요소 및 대기질 정보의 3차원 표출에 대한 원형 시스템을 개발하였으므로 이 프로그램을 점차 발전시켜 기상 및 대기질 정보뿐만 아니라 수문이나 항공 등 다양한 분야로 활용할 수 있을 것으로 기대된다.

감사의 글

이 연구는 2017년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업이다(No.2017R1D1A3B03036152).

REFERENCES

Fuchs, R., Hauser, H., 2009, Visualization of multi-variate scientific data, Computer Graphics forum, 28, 1670-1690.

Im, W. H., Lee, Y. W., Suh, Y. C., 2010, Development of a web-based geovisualization system using google earth and spatial DBMS, Journal of korea spatial information society, 18, 141-149.

IPCC, 2007, Climate change 2007: Synthesis report. Contribution of working groups I, II and III to the fourth assessment report of the intergovernmental panel on climate change, Geneva, Switzerland.

Keqi, Z., Shu-Ching, C., Peter, S., 2006, A 3D visualization system for hurricane storm-surge flooding, IEEE computer society, 18-25.

Kim, H. D., Lee, J. H., 2014, Case studies of predicting volcanic ash by interactive realtime simulator, Journal of environmental science international, 23, 2121-2127.

Roberto, S. J., Juan, L. P., Rosa, M. G., 2011, 3D

Visualization of air quality data, 11th international conference “Reliability and statistics in transportation and communication”, 1-9.

Ryan, B., 2016, Creating a volumetric ray marcher, https://shaderbits.com/blog/creating-volumetric-ray-marcher.

• 김건우, 윈랩 주식회사 대표이사
zero6589@naver.com
• 나하나, 인제대학교 대기환경정보공학과 석사
hana717@nate.com
• 정우식, 인제대학교 대기환경정보공학과 교수
wsjung1@inje.ac.kr