Journal of the KIMST, Vol. 23, No. 5, pp. 494-501, 2020
ISSN 1598-9127(Print) · 2636-0640(Online)
DOI https://doi.org/10.9766/KIMST.2020.23.5.494

Research Paper 정보·통신 부문

폴리곤 분할 기법을 사용한 해안선 지도 전시 성능개선

장택수*,1) · 나범철1) · 홍동호1) · 박근국1)

1) LIG넥스원㈜ 유도무기2연구소

Performance Improvement of Coastline Map Using Polygon Splitting Technique

Taeksoo Jang*,1) · Beomcheol Na1) · Dongho Hong1) · Keunkuk Park1)

1) *Launcher Control Systems for Rockets and Missiles, LIGNex1, Korea*

(Received 6 July 2020 / Revised 10 September 2020 / Accepted 25 September 2020)

Abstract

When we develop an OpenGL-based coastline map that displays the high resolution data of GSHHG, it takes tens of seconds until the first scene is rendered. That is because it takes time to tessellate polygons with a huge number of vertices. A commonly used solution is that it converts original data to a customized data in advance, and it displays these. But if we should use original data to display a coastline map, we need to find another solution. In this paper we suggest using a polygon splitting technique to minimize tessellation time. In order to prove its effectiveness, we implemented a software applied this technique and measured its performance. The software applied this technique took a few seconds to display the first scene and we confirmed its effectiveness.

Key Words : Coastline Map(해안선 지도), Polygon Splitting(폴리곤 분할), GSHHG(고해상도 지리 데이터)

1. 서 론

유도무기 발사통제시스템에서 전술상황 전시를 위해 해안선 지도를 사용하는 경우가 있다. 해안선 지도에서 표시하는 주요 지리정보는 바다, 육지 및 국경선이며, 해안선 데이터를 사용한다. 해안선 데이터로 사용할 수 있는 데이터에는 미국 National Centers for Environmental Information(NCEI)에서 제공하는 Global Self-consistent, Hierarchical, High-resolution Geography Database(GSHHG)[1]가 있다. 이 데이터는 다루기 쉬운 ESRI Shapefile 포맷으로도 배포되는데, 데이터는 주로 다수의 폴리곤과 폴리라인으로 구성된다. 폴리곤 데이터는 전세계 육지와 바다 및 호수의 경계 정보를 나타내고, 폴리라인 데이터는 나라, 주(state) 및 해상 경계를 나타낸다.

OpenGL을 사용하여 해안선 지도를 구현하는 경우, 고해상도(High Resolution) GSHHG 데이터를 로딩하여 전세계를 전시할 때까지 40초가 넘게 걸렸다. 첫 지도 전시까지 너무 오랜 시간이 소요되기 때문에 발사통제시스템에서 사용하기 위해서는 당연히 개선해야 한다. 원인 분석 결과, 소요시간의 대부분은 테셀레이션

* Corresponding author, E-mail: taeksoo.jang@lignex1.com

[2] 작업시간이었다. 테셀레이션은 OpenGL로 면이 채워진 오목한(Concave) 폴리곤을 그리기 위한 필수적인 작업이다. 따라서 전시 성능을 개선하려면 테셀레이션 작업시간을 최소화시켜야 한다.

해안선 지도 구현 시 테셀레이션 소요시간을 줄이기 위해 주로 사용하는 방법은 1) 원본 데이터를 테셀레이션하여 파일로 저장한 후 이를 사용하거나, 2) GIS 프로그램으로 정점개수가 많은 폴리곤을 분할하여 파일로 저장한 후 이를 사용하는 방법이 있다. 하지만 두 경우 모두 원본파일을 미리 변환해서 사용하는 방법이기 때문에, 원본파일을 그대로 사용해야 하는 요구사항이 있다면 새로운 방법을 고안해야 한다.

본 논문에서는 해안선 데이터를 원본 그대로 사용하면서도 지도 전시성능에 문제가 없는 폴리곤 분할 기법 사용을 제안한다. 또한 제시한 방안을 적용한 지도전시 소프트웨어를 구현하고 성능을 측정하여 제안 방안의 개선 정도를 확인한다.

2장에서는 본 논문에서 다루는 해안선 지도에 대한 설명과 성능저하 원인을 분석하고 성능 개선을 위해 검토했던 다른 방안을 간략히 소개한다. 3장에서는 본 논문에서 제안하는 폴리곤 분할 기법을 설명한다. 그리고 4장에서는 개선 전후를 비교할 수 있도록 성능 측정 결과를 제시한 후 5장에서 결론을 맺는다.

2. 해안선 지도

2.1 해안선 지도 형상

본 논문에서 대상으로 삼는 해안선 지도는 Fig. 1과 같은 형상이다.

이 해안선 지도는 일부 유도무기 발사통제시스템에서 사용하는 해안선 지도로 바다, 육지 및 국경선을 표시한다.

Fig. 1. Example of coastline map

2.2 해안선 데이터

해안선 지도 전시를 위한 해안선 데이터는 미국 NCEI에서 GNU LGPL 라이선스로 배포하는 GSHHG를 사용할 수 있다. GSHHG는 World Vector Shorelines (WVS), CIA World Data Bank II(WDBII) 및 Atlas of the Cryosphere(AC)를 통합한 고해상도 지리 데이터이다. 이 데이터는 5가지 해상도로 배포되는데 원본 데이터(Full Resolution)보다 한 단계 낮은 해상도인 고해상도(High Resolution)를 사용하면 확대시에도 적절한 수준의 해안선을 전시할 수 있다.

GSHHG는 ESRI Shapefile 포맷과 바이너리 포맷으로 배포된다. Shapefile 데이터는 해안선(Shoreline) 데이터를 육지, 호수, 호수내 섬, 호수내 섬의 연못, 얼음전선(Ice Front) 기반 남극대륙, 지반선(Grounding Line) 기반 남극대륙과 같이 6가지 레벨로 구분하여 제공한다. 그리고 행정구역(Political Boundary)은 국경, 주 경계, 해양 경계와 같이 3가지 레벨로 구분하여 제공한다. 해안선 지도를 전시하는데 필요한 데이터는 육지 표시를 위한 GSHHS 레벨1 데이터와 국경 표시를 위한 WDBII 레벨1 데이터이다. GSHHG 데이터에서 육지는 폴리곤으로 저장되어 있고 국경선은 폴리라인으로 저장되어 있다.

2.3 해안선 지도 전시

본 논문에서 연구한 해안선 지도는 다음과 같은 제한사항이 있다. 첫째, 해안선 데이터를 원본 그대로 사용하여 지도를 전시해야 한다. 지도 데이터 로딩 및 전시 성능 개선을 위해 원본 데이터를 미리 변환하고 이를 사용하는 전시방법은 제외한다. 둘째, 오목한 폴리곤 전시를 위해 테셀레이션 작업이 필요한 OpenGL이나 Direct3D와 같은 그래픽스 라이브러리를 사용하는 경우다. Windows GDI와 같이 테셀레이션 과정 없이 오목한 폴리곤을 그릴 수 있는 API는 논외로 한다.

해안선 지도는 해안선 데이터 로딩 후 육지와 국경선 그리기를 화면 갱신주기에 따라 반복하여 수행한다. 주기에 따라 반복하여 전시하는 이유는 표적이나 자산과 같은 지도 도시요소의 정보가 일정한 주기로 갱신되기 때문이다.

Fig. 2는 OpenGL을 사용한 해안선 지도 전시 절차를 보여준다. 해안선 데이터 로딩 후 데이터가 폴리곤인 경우 테셀레이션 작업을 수행하고, 폴리라인인 경우 전처리 없이 OpenGL 프리미티브(Primitive)로 그린다. 보통 소프트웨어 응답성을 높이기 위해 별도 스레

드에서 해안선 데이터 로딩과 폴리곤 테셀레이션을 수행한다. 그리고 동일한 데이터를 갱신주기에 맞추어 반복적으로 전시하기 때문에 디스플레이 리스트를 사용하는 것이 효과적이다.

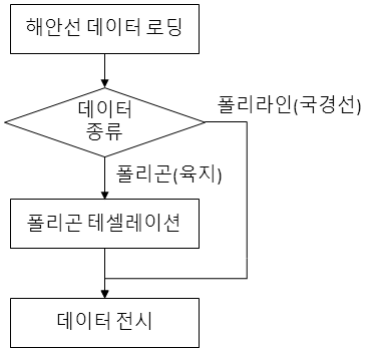


Fig. 2. Basic procedure of displaying coastline map

2.4 해안선 지도 성능저하 원인

2.3절에서 설명한 전시방법으로 해안선 지도를 구현하고 성능을 측정해보았다. 사용한 해안선 데이터는 Table 1과 같이 2개 파일이며, GSHHG 버전 2.3.7의 고해상도(High Resolution) 데이터 파일 중 2개에 해당한다. 성능측정은 첫 전시화면으로 전세계를 표시할 때를 측정하였다. 해안선 지도는 Table 2의 장비에서 실행하였으며 작업 소요시간을 윈도우즈 제공 API인 GetTickCount() 함수로 측정하였다.

Table 1. GSHHG data to draw

파일명(크기)	데이터 정보	
GSHHS_h_L1.shp (33,330KB)	폴리곤	144,749개
	정점	총 1,626,467개
	폴리곤당 정점	▪ 100,000 이상: 2건 ▪ 10,000~100,000: 6건 ▪ 1,000~10,000: 61건 ▪ 4~1,000: 144,680건
WDBII_border_h_L1.shp (1,117KB)	라인	451개
	정점	총 69,886개
	라인당 정점	▪ 2,000 이상: 1건 ▪ 1,000~2,000: 7건 ▪ 2~1,000: 443건

Table 2. Specification of measuring equipment

구분	사양
CPU	Intel(R) Core(TM) i5-9400 CPU@2.9GHz
Memory	16.0GB
GPU	NVIDIA GeForce GTX 750 Ti
OS	Windows 10 Enterprise 2016 LTSB

Table 3. Results Summary

구분		소요시간(초)
첫 지도 전시	해안선 데이터 로딩	0.532
	육지 전시	44.438
	국경선 전시	0.000
이후 지도 전시		0.050

Table 4. Tessellation time

순번	소요시간 (초)	정점 개수	순번	소요시간 (초)	정점 개수
1	23.610	139,786	18	0.016	4,546
2	15.859	111,436	19	0.016	4,238
3	1.032	40,118	20	0.016	4,751
4	0.687	27,578	21	0.016	2,857
5	0.656	28,032	22	0.016	2,724
6	0.578	29,980	23	0.015	5,049
7	0.328	20,054	24	0.015	3,176
8	0.094	10,022	25	0.015	4,687
9	0.047	6,359	26	0.015	3,224
10	0.047	6,515	27	0.000	2,566
11	0.047	7,737	28	0.000	3,187
12	0.031	5,734	29	0.000	2,302
13	0.031	7,198	30	0.000	2,473
14	0.031	5,665	31	0.000	3,089
15	0.031	5,278	32	0.000	2,121
16	0.016	3,752	33	0.000	2,298
17	0.016	5,776	34	0.000	2,401

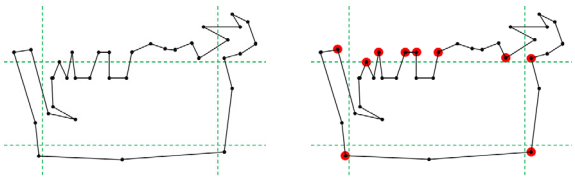
Table 3은 1회 측정한 결과를 구분하여 표시하였다. 데이터 로딩 후 첫 지도 전시까지 약 46초가 걸리고 이후 지도 갱신은 평균 0.050초가 걸렸다. 구현한 해안선 지도에서 성능개선이 필요한 부분은 첫 지도전시에서 육지 전시 부분이다. 육지 전시에서 오랜 시간이 걸리는 이유는 정점개수가 많은 폴리곤의 테셀레이션 작업에서 많은 시간이 걸리기 때문이다.

Table 4는 측정결과를 테셀레이션 소요시간 기준으로 내림차순 정렬한 데이터 중 상위 34개만 추출한 데이터다. 측정결과로 알 수 있는 사실은 지형 데이터의 정점 개수가 10,000개 이하인 경우 테셀레이션 시간이 0.1초 이내라는 것이다. 즉, 폴리곤 하나의 정점 개수를 10,000개 이하로 줄이는 방법을 찾는다면 전시 성능이 상당히 개선될 것이라고 추측할 수 있다.

2.5 성능 개선 방안 검토

해안선 지도 성능개선을 위해 가장 먼저 검토했던 방안은 폴리곤 클리핑이다. 정점개수가 10,000개 이상인 폴리곤들을 대상으로 전시영역을 벗어나는 정점들을 제거한다면 테셀레이션 소요시간이 줄어들 것이기 때문이다. 폴리곤 정점 제거를 위해, 라인(line) 클리핑 알고리즘인 Cohen-Sutherland 알고리즘[3]을 활용하였다. Cohen-Sutherland 알고리즘은 2차원 공간을 전시영역을 중심으로 9개 영역으로 분할하고 라인의 양종단 정점에 대한 outcode 값을 산출하여 전시영역에 포함되는 라인만을 효과적으로 추출한다.

Fig. 3-(1)은 전시영역을 포함한 9개 영역(녹색 점선으로 구분된 영역)과 폴리곤이 가질 수 있는 형태를 종합적으로 나타낸 그림이다.



(1) 전시영역과 폴리곤 (2) 전시를 위한 최소한의 외부 정점 선택

Fig. 3. Minimum selection of vertices outside the viewport

그림은 한 점이 전시영역 경계에 걸치는 경우, 한 점이 전시영역 외부로 나가는 경우, 두 점이 전시영역 외부로 나가는 경우, 두 점 이상이 전시영역 외부로 나가는 경우, 여러 점이 전시영역 외부를 둘러싸는 경우 등을 포함한다. Fig. 3-(2)는 전시영역에 보여지는 폴리곤의 형태를 원래 폴리곤과 동일하게 유지하기 위해 최소한으로 존재해야 하는 외부 정점들을 빨간색 원으로 표시한 그림이다. 즉, 폴리곤 전시 시에 전시영역 안에 있는 모든 정점들과 최소한으로 선택된 외부 정점들만 있다면 전시영역에 보여지는 폴리곤은 원래 형태를 유지한다. Fig. 3-(2)에 선택된 외부 정점들의 공통점은 해당 정점을 포함한 간선이 2개 영역에 걸친다는 것이다. 따라서 다음 두 경우에 해당하는 정점들만을 추출한다면 불필요한 정점을 최대한 제거하고 폴리곤을 그릴 수 있다.

1) 전시 영역 안의 정점
2) 2개 영역에 걸치는 간선의 두 정점

이 방안은 Table 2의 장비에서 100 km × 100 km 수준의 영역을 전시할 경우 CPU 사용률이 4 % 수준이고 한반도 전체를 전시하는 경우 CPU 사용률이 14 % 수준이었다. 그러나 전세계를 전시하는 경우에는 전시영역을 벗어나는 정점이 없기 때문에 테셀레이션 소요시간이 전혀 줄어들지 않는다. 따라서 이 방안은 지도 전시 영역이 좁은 경우에만 적합하기 때문에 보편적으로 사용하기에 부적합하다.

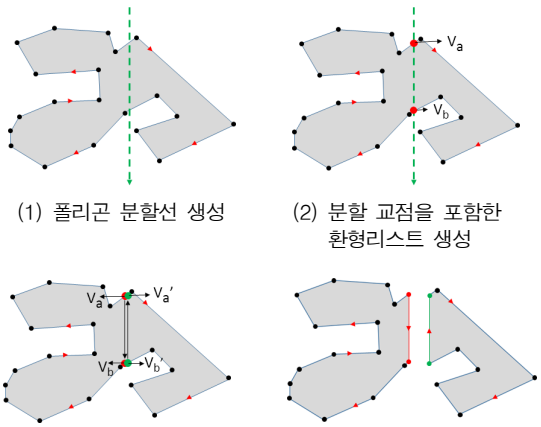
다음으로 검토한 방안은 폴리곤 분할 기법[4,5]을 적용하는 것이다. 정점개수가 10,000개 이상인 폴리곤을 분할하여 각 폴리곤의 정점개수를 10,000개 이하로 줄인다면 테셀레이션 소요시간도 줄어들 것이기 때문이다. 참고문헌 [4]는 3차원 공간의 폴리곤을 평면으로 분할하는 폴리곤 분할 기법이고, 참고문헌 [5]는 2차원 상의 폴리곤을 폴리라인으로 분할하는 기법이다. 본 연구의 목적은 폴리곤의 정점개수가 10,000개 이하가 되도록 반복적으로 분할하는 것이므로, 적용하기 쉬운 참고문헌 [4]를 사용하여 폴리곤 분할을 수행하였으며 적용한 내용은 3장에서 자세하게 설명한다.

3. 해안선 지도 성능개선 방안

본 논문에서 제안하는 해안선 지도 전시 개선방안은 Fig. 2의 폴리곤 테셀레이션 단계 전에 폴리곤 분할 단계를 추가하는 것이다. 폴리곤 분할은 폴리곤의 정점 개수가 10,000개 이하가 될 때까지 반복하여 분

할한다. 적용한 폴리곤 분할 기법은 3차원 공간의 폴리곤을 평면으로 분할하는 폴리곤 분할 기법[4]을 응용하여 2차원 상의 폴리곤을 세로방향 직선으로 분할한다. 이때 세로방향 직선의 위치는 폴리곤의 가로 영역을 이등분하는 위치가 된다. 기존 기법과 논문에서 적용한 기법의 차이는 차원(3차원, 2차원)과 폴리곤을 구성하는 정점들의 순서(counter-clockwise, clockwise)에 따른 차이가 존재한다.

Fig. 4는 폴리곤을 1회 분할하는 과정을 순차적으로 표현한다. Fig. 4에서 세로방향 녹색점선은 폴리곤 분할선을 의미하며, V_a와 V_b는 분할선과 폴리곤의 교점을, V_a'와 V_b'는 분할 간선 추가를 위해 교점을 복사하여 추가한 정점을 의미한다.



(1) 폴리곤 분할선 생성 (2) 분할 교점을 포함한 환형리스트 생성

(3) 분할 간선 추가 (4) 폴리곤별 정점 목록화

Fig. 4. Polygon splitting process

Fig. 4-(4)와 같이 분할된 폴리곤들을 표현할 수 있도록, 폴리곤 정점을 이중 연결 리스트로 표현하여 저장한다. 이를 위한 폴리곤 정점 정보 구조체는 다음과 같다.

```
struct PolygonVertex
{
    CPointF    vertexPos;     //정점좌표 x, y, z
    VertexSide vertexSide;    //분할선 기준 정점위치
    PolygonVertex*  pNext; //시계방향 다음 정점
    PolygonVertex*  pPrev; //시계방향 이전 정점
    float      fDistFromRefP;  //분할 기준점과의 거리
    bool       bCollected; //폴리곤 정점 수집용 플래그
};
```

vertexPos 필드는 정점좌표 (x, y, z)이며 x는 경도, y는 위도, z는 고도(고정값 0)이다. vertexSide 필드는 분할선을 기준으로 정점 위치를 왼쪽(Left), 위(On), 오른쪽(Right) 중 하나로 나타낸다. pNext와 pPrev 필드는 시계방향으로 다음과 이전 정점의 주소 정보를 가진다. fDistFromRefP 필드는 분할선 최상단 정점과의 거리값을 가진다. bCollected 필드는 폴리곤 분할 후 각 폴리곤의 정점을 목록화할 때 해당 정점이 이미 목록화되었는지 아닌지를 나타낸다. 폴리곤을 1회 분할하는 과정은 하위 절과 같다.

3.1 폴리곤 분할선 생성

폴리곤을 세로로 분할하는 선분의 두 정점 좌표를 생성한다. 두 정점의 x좌표는 폴리곤 가로 영역의 중간지점 좌표를 사용하고, y좌표는 위도 범위인 +90.0에서 –90.0을 포함하도록 +100.0과 –100.0을 사용한다. 따라서 폴리곤은 무조건 2개 이상으로 분리된다.

3.2 분할 교점을 포함한 환형리스트 생성

폴리곤과 분할선의 교점(V_a, V_b)을 산출하고 폴리곤의 모든 정점과 산출한 교점을 하나의 list 변수(이하 PreparedPolygon)에 시계방향 순서로 저장한다. 그리고 PreparedPolygon 변수에 저장한 폴리곤과 분할선의 모든 교점의 포인터(메모리 주소)를 별도의 배열(이하 VertexOnSplitline)로 저장한다. PreparedPolygon에 저장되는 모든 정점은 PolygonVertex 구조체로 표현되며 Fig. 4-(2)와 같이 어느 정점에서 시작해도 한 바퀴 순회할 수 있도록 연결정보를 저장한다. 이 단계에서 PolygonVertex 인스턴스에 미지정된 정보는 fDistFromRefP 뿐이며 bCollected는 false로 초기화한다.

3.3 분할 간선 추가

VertexOnSplitline에 저장된 교점 정보를 이용하여 폴리곤 분할을 위한 모든 간선을 추가한다. Fig. 4는 폴리곤이 2개로 분리되는 예이므로 폴리곤을 분할하는 간선의 정점 추출이 간단하다. 하지만 폴리곤과 분할선의 교점을 단순히 2개씩 쌍으로 추출할 수 없는 경우들이 많기 때문에, 간선 추가를 위한 정점 쌍을 식별하는 규칙이 필요하다.

우선 폴리곤과 분할선이 만나는 형태는 Fig. 5와 같이 20가지가 된다. 폴리곤과 분할선이 만나는 형태는 시계방향으로 연속한 3개 정점의 위치를 나타내는 두 문자를 사용하여 명명하였다. 사용한 두문자는 분할선

을 기준으로 폴리곤의 정점이 왼쪽에 있는 경우 'L', 위에 있는 경우 'O', 오른쪽에 있는 경우 'R'을 사용하였다.

Fig. 5의 여러 가지 구성 중에 분할 간선의 source 정점이 될 수 있는 경우는 Fig. 6과 같은 5가지 경우만 존재한다. 이 5가지 경우를 설명하면 다음과 같다.

1) LOR 구성의 정점인 경우다.
2) ROR 또는 LOL 구성의 정점인 경우로 다음 조건을 만족하면 source 정점이 된다. 조건은 이전 분할 간선 추출 시 destination 정점으로 사용한 경우다. 즉, 분할 간선 추출 시, ROR 또는 LOL 구성의 정점을 destination 정점으로 사용했다면, 다음 분할 간선의 source 정점으로 사용하면 된다.
3) '↳' 형태의 OOR 구성 정점인 경우다. 이 형태를 판단하는 방법은 전 정점(pPrev)의 fDistFromRefP 보다 현재 정점의 fDistFromRefP가 더 큰가를 확인하면 된다.
4) '⮥' 형태의 LOO 구성 정점인 경우다. 이 형태를 확인하는 방법도 위와 같이 fDistFromRefP 정보를 이용한다.

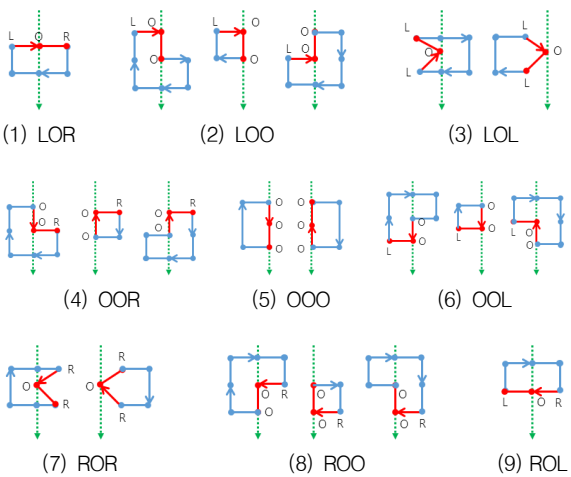


Fig. 5. Possible configurations

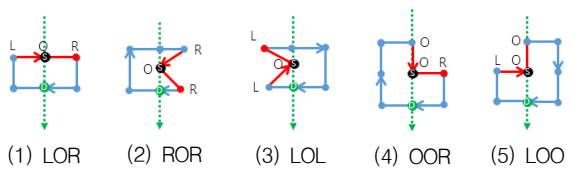


Fig. 6. Configurations as source vertex

Fig. 7은 분할 간선의 destination 정점이 될 수 있는 5가지 경우로 이를 정리하면 다음과 같다.

1) ROL 구성의 정점인 경우다.
2) ROR 또는 LOL 구성의 정점인 경우다.
3) '⮧' 형태의 ROO 구성 정점인 경우로 다음 조건을 만족하면 destination 정점이 된다. 이 형태를 판단하는 방법은 현재 정점의 fDistFromRefP가 후 정점(pNext)의 fDistFromRefP 보다 더 작음을 확인하면 된다.
4) '↰' 형태의 OOL 구성 정점인 경우다. 이 형태를 확인하는 방법도 위와 같이 fDistFromRefP 정보를 이용한다.

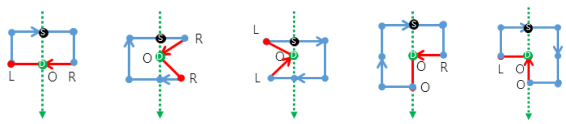


Fig. 7. Configurations as destination vertex

폴리곤을 분할하는 간선 추가를 위한 두 정점을 식별했다면, 분할된 폴리곤의 정점들이 순환하도록 간선을 재구성해야 한다. Fig. 8에서 srcVertex와 destVertex는 PreparedPolygon 변수에 3.2절의 절차에서 이미 추가했던 정점이고, srcVertexNew와 destVertexNew는 분할 간선 추가를 위해 PreparedPolygon 변수에 새로 추가한 정점이다. 정점들 간의 전/후 정보 설정은 그림과 같다.

Fig. 8. Bridge addition

3.4 폴리곤별 정점 목록화

분할이 완료되었으므로 폴리곤별로 정점들을 목록화해야 한다. PreparedPolygon 변수에 저장된 정점들은

Fig. 4 - (4)와 같이 폴리곤별로 순환하도록 되어 있다. 따라서 PreparedPolygon 변수의 첫 정점부터 순차적으로 순환하면서 각 폴리곤의 정점을 목록화하면 된다. 이때 폴리곤의 정점으로 추출된 정점은 bCollected를 true로 설정하여 이미 추출된 정점임을 표시하여 다음 폴리곤 추출에서 제외하도록 한다.

4. 해안선 지도 개선방안 성능확인

해안선 지도 전시성능의 개선 정도를 확인하기 위해, 폴리곤 분할 기법을 미적용한 지도와 적용한 지도를 구현하였다. 성능측정에 앞서 폴리곤 분할이 정확하게 수행되었는지 확인하였다. 우선 모든 폴리곤의 정점이 10,000개 이하로 분할되었는지 확인하였다. 분할 과정을 로깅하여 분석한 결과, 폴리곤의 정점개수가 10,000개를 넘어서 분할이 발생한 횟수는 총 45회였고, 폴리곤 분할결과를 집계한 데이터는 Table 5와 같다. 분할 결과 데이터를 Table 1과 비교했을 때 폴리곤의 총 개수는 283개가 늘어났고 폴리곤의 총 정점개수는 1,415개 늘어났다. 분할결과 증가된 폴리곤 개수는 283개인데 증가된 총 정점개수는 1,415개로 4배가 아닌 5배가 된 이유는 Shapefile의 폴리곤 조건을 맞추기 위해 분할된 폴리곤 2개 중 1개에 정점 1개를 더 추가했기 때문이다. Shapefile의 폴리곤 조건 중 '폴리곤은 닫힌(closed) 상태가 되어야 한다'는 것이 있는데, 닫힌 상태란 폴리곤의 시작 정점과 끝 정점이 같다는 것을 의미한다.

Table 5. Polygon splitting result

파일명(크기)	데이터 정보	
GSHHS_h_L1.shp (33,330KB)	폴리곤	145,032개
	정점	총 1,627,882개
	폴리곤당 정점	▪ 1,000~10,000: 135건 ▪ 4~1,000: 144,897건

이번에는 분할된 폴리곤들의 합이 이전 모양과 동일한지 확인하기 위해 폴리곤 분할선과 분할된 폴리곤들을 Fig. 9와 같이 가시화하였다. 가시화는 폴리곤 분할선들을 녹색으로 표시하고 분할된 폴리곤들도 분할한 경계가 확인되도록 서로 다른 색상을 사용하여 전시하였다. 이후 상용 지도 소프트웨어를 사용하여 분할된 영역을 확대하여 비교함으로써 폴리곤 분할이 정확하게 수행되었음을 확인하였다.

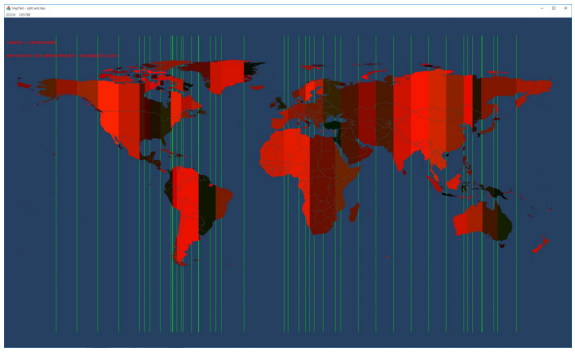


Fig. 9. Visualization of polygon splitting result

Table 6은 분할된 폴리곤들의 테셀레이션 소요시간을 내림차순으로 정렬한 후 상위 34개만을 추출한 데이터다. 예상했던 것과 같이 모두 0.1초 이내에 테셀레이션을 완료하였다.

Table 6. Tessellation time of splitted polygons

순번	소요시간 (초)	정점 개수	순번	소요시간 (초)	정점 개수
1	0.078	9,370	18	0.047	6,515
2	0.078	9,326	19	0.047	6,136
3	0.063	8,229	20	0.047	5,734
4	0.063	7,956	21	0.046	8,022
5	0.063	7,414	22	0.046	7,198
6	0.063	7,043	23	0.032	6,703
7	0.062	9,936	24	0.032	6,017
8	0.062	9,723	25	0.032	5,791
9	0.062	8,681	26	0.032	5,521
10	0.062	8,587	27	0.031	7,101
11	0.062	7,864	28	0.031	7,017
12	0.062	7,290	29	0.031	6,862
13	0.047	9,015	30	0.031	6,851
14	0.047	8,099	31	0.031	6,572
15	0.047	7,737	32	0.031	6,478
16	0.047	7,653	33	0.031	6,146
17	0.047	6,768	34	0.031	4,899

전시성능 측정은 Table 2의 장비를 사용하여 첫 지도 전시화면에 1) 전세계를 전시하는 경우와 2) 좁은 작전영역 수준인 1° × 1° 영역을 전시하는 경우로 구분하여 측정하였다. 1° × 1° 영역은 전시하는 위치에 따라 소요시간이 많이 차이나기 때문에, 많은 정점수를 가지는 아시아 대륙과 여러 섬들이 전시되는 한반도 서해안 지역(중심좌표 126.4890°E, 36.5980°N)으로 선정하여 측정했다.

Table 7. Performance measurement

전시 방법	전시 영역	소요시간(sec)		
		데이터 로딩	폴리곤 분할	첫 지도 전시
폴리곤 미분할	1° × 1°	0.537	-	24.130
	전세계	0.567	-	45.381
폴리곤 분할	1° × 1°	0.552	0.198	1.109
	전세계	0.557	0.193	4.781

Table 7의 소요시간은 3회 측정한 값을 평균한 것이다. 소요시간은 데이터 로딩, 폴리곤 분할, 첫 지도전시로 구분하여 측정하였다. 당연하게도 데이터 로딩 시간은 폴리곤 분할과 관계없이 비슷한 수준의 소요시간이 걸린다. 차이는 첫 지도전시 소요시간으로 개선한 지도는 전세계 전시 시에 약 5초가 걸렸다. 또한 1° × 1°의 영역을 표시하는 경우 약 1초만에 지도를 전시하였다.

5. 결 론

OpenGL 기반 해안선 지도 개발 시, 데이터 해상도가 높은 경우 초기 지도 전시까지 수십초가 걸리는 문제가 있다. 원인은 정점 개수가 많은 폴리곤을 테셀레이션하는데 많은 시간이 소요된다는 것이다. 이에 대한 해결책으로, 본 논문은 폴리곤 분할 기법을 사용하여 개별 폴리곤의 정점 개수를 줄이는 방안을 제안하였다. 또한 제안한 방법이 실제로 효과가 있는지 확인하기 위해, 소프트웨어를 구현하고 성능을 측정하여 개선 정도를 확인하였다. 성능측정 결과에서 확인한 것과 같이 제안한 방법은 초기 전시 소요시간을 크게 줄였으며, 신속한 전개가 필요한 무기체계에 적용하기에 무리가 없는 수준임을 확인하였다.

신규 무기체계에 사용할 해안선 지도를 개발할 때, 본 논문에서 제안한 방법을 사용한다면 개발 시간과 비용이 감소할 것으로 기대된다.

References

[1] Paul Wessel, Walter H. F. Smith, “A Global, Self-consistent, Hierarchical, High-resolution Shoreline Database,” Journal of Geophysical Research, Vol. 101, pp. 8714-8743, 1996.

[2] Dave Shreiner, Gihyuk Nam(Trans.), “OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.4,” Information Publishing Group, Korea, pp. 519-538, 2005.

[3] James D. Foley, “Computer Graphics: Principles and Practice,” Addison-Wesley Professional, United States, p. 113, 1996.

[4] Alan W. Paeth, “Graphics Gems V,” Academic Press, United States, pp. 386-393, 1995.

[5] L. Wu, G. Tian and Z. Xie, “An Algorithm for Splitting Arbitrary Polygon,” 2009 Fifth International Conference on Natural Computation, pp. 318-325, 2009.