Journal of Korean Institute of Intelligent Systems, Vol. 22, No. 3, June 2012, pp. 334-340
http://dx.doi.org/10.5391/JKIIS.2012.22.3.334

임베디드 시스템에서의 전자해도 표출 방안 연구

Study on How to Display S-57 ENCs in a Embeded Mobile-Platform

조기정* · 이주환**†
Gi-jong Jo, Ju-Hwan Lee†

*(주)지엠티 기술연구소, 한국해양대학교 해양교통학과
**(주)지엠티 기술연구소

요 약

본 연구는 SOLAS의 ECDIS 강제 탑재규정에 제외되는 소형선에서의 모바일플랫폼 및 저사양 단말기를 이용한 S-57 전자해도 데이터(ENC) 활용을 위해 Tiled-SENC 포맷을 정의하고, ENC에서 Tiled-SENC로의 자료구조 모델링 및 변환 기술를 개발하고, ENC와 Tiled-SENC 사용시의 성능을 비교하였다.

키워드 : Tiled-SENC, ENC Converter, GPS-Plotter, 소형선용 ECS

Abstract

This project defines Tiled-SENC format for S-57 ENC data usage of the mobile platforms and low-end terminals which are not included in ECDIS carriage requirements of SOLAS Chapter V. Also, the architecture modeling and converting technology of Tiled-SENC have been developed and the performance has been compared with ENC Format.

Key word : Tiled-SENC, ENC Converter, GPS-Plotter, Mobile Platform ECS

1. 서 론

기존의 대형선이나 해상교통관제센터에서 사용하는 전자해도표출시스템은 기본적으로 고성능의 처리능력을 갖는 시스템을 요구하므로 어선이나 레저용 보트와 같은 소형선에는 주로 사용하는 PDA, 웹패드와 같은 임베디드 기반 모바일 플랫폼에서 해상 지형정보, 특히 국가공인 데이터인 전자해도를 사용하기 위한 방안이 요구되고 있고 여러기관에서 관련 연구를 수행하고 있다[1].

본 연구에서는 모바일 장치에서 사용할 있도록 소형화된 전자해도 기술 확보를 위해, 저사양(메모리, CPU) 단말기에서 제한된 자원을 이용한 실시간 ENC Converter 구조 및 저사양에 적합한 Tiled-SENC 모델을 정의하고, 필요한 Data부분을 특정영역에서 Random Access하기 위한 Indexing구조를 개발하였다. 또한 공인 전자해도로부터 Tiled-SENC로의 변환기를 개발하여 저사양 모바일 단말에서의 사용 가능성을 검증하였다.

접수일자: 2012년 4월 20일
심사(수정)일자: 2012년 6월 4일
게재확정일자 : 2012년 6월 4일
† 교신 저자
본 연구는 2011년 지식경제부 해양레저장비산업 경쟁력 강화 사업, 과제번호: K0005827에 의해 수행되었습니다, 연구비 지원에 감사드립니다.
본 논문은 본 학회 2012년도 춘계학술대회에서 선정된 우수논문입니다.

2. S-57 전자해도 모델

그림 1과 2와 같이 S-57 국제표준의 데이터 모델은 피쳐(Feature)정보와 공간정보로 구성되며 상호 연결된다. Feature정보는 독립적으로 존재할수 있으나 공간 정보는 Feature 정보에 연계되어 존재할 수 있다. 그리고 공간정보에는 점, 선, 면에 해당되는 노드(Node), 에지(Edge), 페이스(Face)로 구분되며, 노드로는 독립 노드, 연결 노드로 구성된다. 또한 전자해도는 공간 모델에서 점, 선에 해당하는 노드와 에지 정보를 이용하여 모든 공간정보를 표현하며, 면에 해당 정보는 에지에 연결하여 표현할 수 있다.

S-57 표준에서 수로데이터는 레코드 단위로 저장되며, 레코드에는 필드, 필드에는 다시 부필드(SubField)로 구성된다. 수로데이터를 파일로 간주할 수 있으며, 서비스를 위해 복수개의 파일과 각 파일에 대한 메타정보를 포함한 교환 셋으로 구성된다[2].

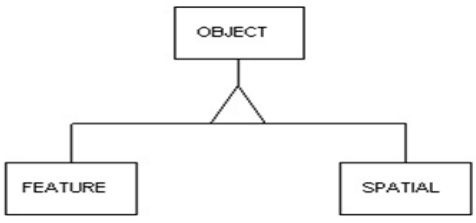

그림 1. S-57 객체모델
Fig. 1. S-57 Object Model

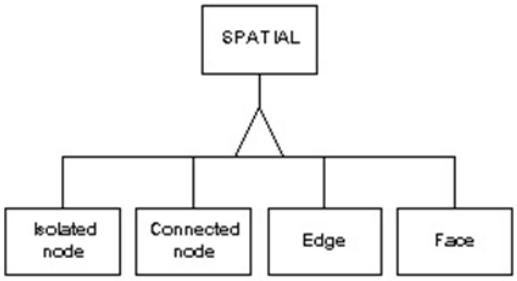

그림 2. S-57 벡터모델
Fig. 2. S-57 Vector Model

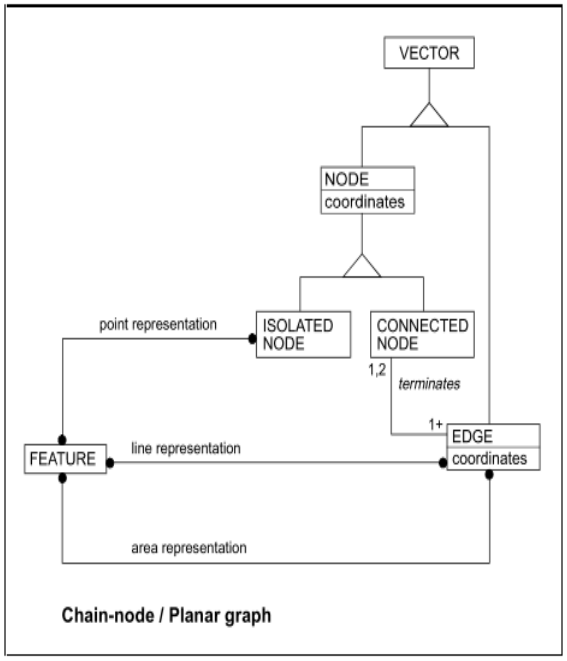

그림 3. S-57 벡터모델
Fig. 3. S-57 Vector Model

각각의 Vector Record는 아래 그림 4와 같이 한건의 Spatial Data를 가질수도 있지만, 여러건의 Vector Record가 모여 하나의 Vector Record를 이루기도 하고, Vector Record와 Spatial Data가 조합되기도 한다. Feature Record 역시 하나의 Vector Record 또는 복수의 Vector Record의 조합으로 이루어진다.

Field Tag	Field Tag	Field Tag	Field Tag
VRID	SG3D		
VRID	SG3D		
VRID	SG2D		
VRID	VRPT	VRPT	
VRID	VRPT	VRPT	SG2D
...	...	...	...
FRID	FSPT	FSPT	FSPT ...
FRID	FSPT	FSPT	FSPT ...

그림 4. S-57 DR Data
Fig. 4. S-57 DR Data

각 Record의 적재 순서는 전체적으로, Vector, Feature순으로 적재되며, Vector는 개별 Spatial, Spatial의 조합 또는 Vector와 Spatial조합, Vector의 조합 순으로 적재되므로, 상위 Vector정보로 하위 Vector를 구성할 수 있는 구조이다.

Feature는 각각의 Vector Record 또는 이들의 조합으로 이루어져 있으므로, 이 구조를 이용하여 전체 ENC File을 스캔하지 않고도 개별 Feature Record를 구성할 수 있어, 제한된 자원을 이용한 실시간 Converter 개발이 가능할 수 있었다.

각각의 Field는 여러개의 SubField를 가지며, 각각의 SubField의 구성형태는 S-57 파일의 DDR Reader부분에 명시 되어 있으며, ENC Product Specification에 규정되어 있고, 그 예시는 아래 그림 4와 같다.

VRID	SG2D	SG3D	VRPT	FRID	FSPT
RCNM	YCOO	YCOO	NAME	RCNM	NAME
RCID	XCOO	XCOO	ORNT	RCID	ORNT
RVER		VE3D	USAG	PRIM	USAG
RUIN			TOPI	GRUP	MASK
			MASK	OBJL	
				RVER	
				RUIN	

그림 5. S-57 DR Data 예시
Fig. 5. Example of S-57 DR Data

3. Tiled-SENC 모델링과 변환방안

3.1 Tiled-SENC 데이터구조 모델링

본 연구를 통해 제시하는 Tiled-SENC 모델은 기존 전자해도의 chain-node 형태와는 달리 full-topology를 적용하여, 실시간 처리를 줄이고, 전체 전자해도 데이터를 적당한 크기의 격자로 나누어 분할 적재하고, 특정 영역에서 Random Access가 가능하도록 Indexing구조를 제공하므로 써, 필요한 소량의 Data만 즉시 메모리에 적재하여 저 사양 모바일 단말에서, 메모리 사용량은 물론 화면표출에도 적은 량의 데이터만 처리할 수 있는 구조로 모델링 하였다.

ENC의 Feature Record를 하나의 객체로 규정하고, 각 객체에 대한 전처리 가능한 부분(Line Style, Fill Pattern, Text 표시 방법등)을 전처리하여, 단말에서의 처리량을 최소화하는 방안을 강구하였고, 각 객체의 영역을 추가로 구성하여, 실제 Display되는 객체의 수도 최소화하는 구조로 설계하였다.

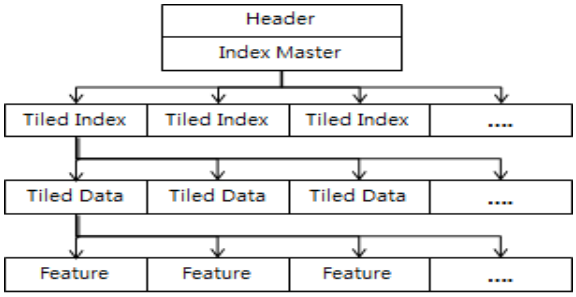

그림 6. Tiled-SENC 데이터 구조
Fig. 6. Tiled-SENC data structure

파일의 헤더 부분에, Level 개수, 각 Level별 구성단위, Scale Min/Max값, Tiled Index 위치등을 기록하여, 파일 전체에 대한 구성을 미리 확인할 수 있게하였다.

Index Master부분에는 각 Level별 Sub-Index부분을 표시하고, Tiled-Index부분은 실제 해당하는 Tile의 위치 및 Tile의 크기를 표시하여 필요시 실제 해당 Level에서 필요한 Tile의 위치로 직접 옮겨갈 수 있고, 필요한 Tile의 Data만 Load 가능한 구조이다. 그 논리적 구성은 아래와 같다.

Boundary	DOUBLE
	DOUBLE
	DOUBLE
	DOUBLE
Max TileSize	UInt32
Max Vert	UInt16
Tile StartX	Int16
Tile StartY	Int16
Tile XCount	Int16
Tile YCount	Int16
Offset	UInt32
Tile Size	UInt32
Obj.Count	UInt16
Obj.Bound	Int32
	Int32
	Int32
	Int32
nVert	UInt16
Point	Int32 * 2

그림 7. Tiled-SENC File Format
Fig. 7. Tiled-SENC File Format

3.2 Tiled-SENC Index Master 구조

Index Master는 그림5와 같이 Level별 Sub-타일을 조회하기 위한 인덱스를 저장하며, 특정 Level의 타일 인덱스값을 효율적으로 검색할 수 있는 구조를 제공한다.

3.3 Tiled Index 구조

Tiled Index는 그림5와 같이 세부 타일을 조회하기 위한 인덱스를 저장하며, 특정영역의 타일 인덱스값을 직접적으로 검색할 수 있는 구조를 제공한다.

3.4 S-57에서 Tiled-SENC 변환절차

본 연구에서는 기존의 해양조사원에서 제공하는 S-57 파일 형식의 표준데이터를 Tiled-SENC로 변환하기 위해 다음의 절차를 수행하였다.

1. 전자해도 로딩 및 파싱
2. 전자해도 - Shape File 변환기
3. Shape File 로딩 및 파싱
4. Tiling Tool 및 SENC 적재 프로그램순으로 개발

우선 전자해도를 로딩하면서, 전체 vector record의 key-field(FSPT, Feature Record To Spatial Recor Pointer Field의 NAME)와, 해당 위치를 보관하고, feature record를 분석하여, 분석 정보를 바탕으로, 각 feature의 full-topology vector data를 구성하였다. 구성된 정보를 shp/shx file로 적재하면서, feature의 attr정보들을 dbf file로 동시에 변환하여, 한 벌의 shape file을 완성하였다. 이렇게 변환된 shape file은 필요에 따라 다시 merge하여 ENC의 각 level별 한 벌의 shape file로 작성하여, tiling tool의 입력 Data는 물론 변환 검증용으로 사용하였다.

변환시 일차적으로 Shape파일로 변환한 이유는, 실시간 Converter를 이용하여, 구성된 각각의 Feature정보를 Tiling하면, 하나의 Tile에 속하는 모든 정보를 제한된 자원상에서 저장하지 않고 메모리에만 적재하기가 한계가 있었고, 일차 변환된 결과를 검증하기위한 별도의 프로그램을 개발해야하는등의 어려움이 있어, Shape File로 일차 저장하였다. Shape File은 많은 여타 Tool에서 검증이 가능하고, 그 사용성이 높아 전달용 포맷으로도 적절하다고 판단하였다. 단, Shape File은 하나의 파일에 동일 종류(점/선/면)의 객체만 저장 가능하여, 저장시에는 각 Feature Type별로 Point/Line/Area 3개의 파일로 저장하였다.

Shape File은 공간 형상을 표현, 설명하기 위해 비위상구조를 가지지 않는 기하학 정보와 속성정보를 저장한다. 공간 형상(feature)의 기하학 정보는 일련의 벡터 좌표들로 구성되는 shape에 저장된다. 위상구조를 가지지 않기 때문에 빠르게 공간 데이타를 디스플레이, 에디팅이 가능하다. Shapefile은 비연속적이거나 중첩되는 공간 형상 또한 손쉽게 다룰수 있다. 또한 디스크 용량을 감소시키며, 데이타를 읽고, 쓰는데 용이하다. Shape File은 point, line, area feature를 지원하며, 속성으로는 dBASE 포맷 파일을 가진다. 각 속성 레코드는 관련된 shape 레코드와 일대일 관계를 가진다.

Shape File은 main file, index file, dBASE table로 구성된다. Main file은 직접 억세스(direct access)가 가능하고, 가변길이 레코드를 포함하며, 이들 가변길이 레코드는 vertex들을 가진 shape를 기술,설명한다. 그리고 인덱스 파일의 각 레코드는 main file의 해당 main file record의 offset를 포함한다. DBASE table은 공간형상 당 하나의 레코드를 가진 속성을 포함한다. 기하학과 속성의 일대일 관계를 가지며, DBASE 파일의 속성 레코드는 main file의 레코드들과 같은 순서를 가져야 한다.

main file은 고정 길이의 파일 헤더 부분이 맨 처음 나오고, 그다음에 가변길이의 레코드들이 리스트된다. 그리고 각 가변 길이 레코드는 고정 길이 레코드 헤더의 내용에 따라 정보를 저장한다.

인덱스 파일은 100바이트 길이의 헤더부분을 가지며, 헤더 부분 다음에 8바이트의 고정 길이 레코드들이 온다.

dBASE 파일은 다른 테이블과 합쳐질 수 있는 바람직한 feature attribute와 속성 키를 포함한다. 그 포맷은 Window와 DOS의 많은 table 어플리케이션에서 사용되는 표준 DBF 포맷이다. 테이블은 shape feature당 하나의 레코드를 포함해야 하고, 레코드 순서는 main file(.shp)에서 shape feature의 순서와 같아야 한다.

각 shp파일은 여러 가지 형태가 있는데, SG3D,

Line/Area의 Mask속성을 추가하기 위해 PointZ, PolyLineZ, PolygonZ 포맷을 사용하였다[5].

Shape File 및 DBF File 변환시 사용된 자료구조는 아래와 같이 각 Data의 속성을 표시하는 nType과, 각 type에 적합한 구조를 가진다. 각 구조체는 기본 생성자와 소멸자를 가짐으로써 메모리누수를 방지하도록 설계되었고, 저사양 단말에서의 메모리 누수에 의한 프로그램 오작동을 고려하였다.

수심을 고려한 Z/Y/Z형 자료를 가지는 Point 객체.

```
struct SHP_Point_Z_Tag {
    UInt32      nType_;
    ShpPOINT_Z  Pt_;
    SHP_Point_Z_Tag() {
        nType_ = e_POINT_Z;
    }
};
```

각 Line의 Mask속성 처리를 위한 X/Y/Z형 자료이며, 여러개의 Line을 가질수있도록 Part를 관리하며, 전체 객체의 영역을 관리하는 Box를 가지는 Line 객체.

```
struct SHP_Polyline_Z_Tag {
    UInt32      nType_;
    ShpBOX      shpBox_;
    UInt32      nNumPart_;
    UInt32      nNumPoint_;
    UInt32*     pParts_;
    ShpPOINT*   pPt_;
    DOUBLE      dZMin;
    DOUBLE      dZMax;
    DOUBLE*     pZArr_;
    DOUBLE      dMMin;
    DOUBLE      dMMax;
    DOUBLE*     pMArr_;
    SHP_Polyline_Z_Tag() {
        memset(this, 0,
sizeof(SHP_Polyline_Z_Tag));
        nType_ = e_POLYLINE_Z;
    }
    void
    clear() {
        if (pParts_) {
            delete [] pParts_;
            pParts_ = 0;
        }
        if (pPt_) {
            delete [] pPt_;
            pPt_ = 0;
        }
        if (pZArr_) {
            delete [] pZArr_;
            pZArr_ = 0;
        }
        if (pMArr_) {
            delete [] pMArr_;
            pMArr_ = 0;
        }
    }
};
```

기본적으로 Line 객체와 동일한 구조이며, Type만 Area로 관리되는 Area 객체.

```
struct SHP_Polygon_Z_Tag {
    UInt32      nType_;
    ShpBOX      shpBox_;
    UInt32      nNumPart_;
    UInt32      nNumPoint_;
    UInt32*     pParts_;
    ShpPOINT*   pPt_;
    DOUBLE      dZMin;
    DOUBLE      dZMax;
    DOUBLE*     pZArr_;
    DOUBLE      dMMin;
    DOUBLE      dMMax;
    DOUBLE*     pMArr_;
    SHP_Polygon_Z_Tag() {
        memset(this, 0,
sizeof(SHP_Polygon_Z_Tag));
        nType_ = e_POLYGON_Z;
    }
    void
    clear() {
        if (pParts_) {
            delete [] pParts_;
            pParts_ = 0;
        }
        if (pPt_) {
            delete [] pPt_;
            pPt_ = 0;
        }
        if (pZArr_) {
            delete [] pZArr_;
            pZArr_ = 0;
        }
        if (pMArr_) {
            delete [] pMArr_;
            pMArr_ = 0;
        }
    }
};
```

DBF File의 개별 Field를 관리하기 위한 객체로, 필드명, 필드 Type, 필드의 길이를 가지는 Field 객체.

```
struct DBF_FieldInfo_Tag {
    Int8        FieldName_[11];
    DBF_Field   FieldType_;
    UInt8       FieldLen_;
    UInt8       FieldDecimal_;
```

```
};
```

DBF File의 개별 Record를 관리하기 위한 객체로, 필드 개수와, 이에 매칭되는 char**를 가진다.

```
struct DBF_RowInfo_Tag {
    UInt16          nColumns;
    Int8**          pValues;
};
```

Clipping 알고리즘으로는 대표적으로, Sutherland-Hodgman clipping algorithms, Weiler-Atherton clipping algorithms, Vatti clipping algorithms이 있는데, 제한적인 상위 2가지 알고리즘에 비해, Vatti 알고리즘은 hole을 포함한 complex(self-intersecting) polygon도 처리가 가능하여 이 알고리즘을 적용하였다[6].

이 알고리즘 구현에 사용된 자료구조는 아래와 같다.

각 Vertex의 정보를 관리하는 객체이며, Clipping시 동일한 좌표인지를 확인할수 있는 ==연산자를 지원한다.

```
struct ClipVertex {
    DOUBLE      dX;
    DOUBLE      dY;
    const bool
    operator == (const ClipVertex& rhs) const {
        return (DBL_EQ(dX, rhs.dX)
            && DBL_EQ(dY, rhs.dY));
    }
};
```

개별 Line 또는 Area를 관리하는 객체로, 전체 Vertex의 개수와, 개별 Vertext의 집합체로 이루어 진다.

```
struct ClipVertex_List {
    Int32       nCount;
    ClipVertex* pVertex;
};
```

최종적인 Lie/Area 객체를 관리하는 객체로, 개별 Line/Area의 집합을 표현한다. 개별 Line/Area의 개수와 각 개별 Area의 상태가 Inner(Hole)인지를 관리하는 char* pbHole과, 개별 Line/Area를 관리하는 ClipVertex_List* 로 이루어져 있다. Clipping시 대상 원본 객체, Clipping 될 영역, 그리고, 이2개가 조합된 Clipping 결과를 관리하는 객체이다.

```
struct CClipPolygon {
    UInt32              nPart;
    UInt8*              pbHole;
    ClipVertex_List*    pVertexList;
};
```

최종적으로 CClipPolygon구조 2개를 입력으로 받아, Clipping된 결과가 나온다. 그 결과중 pbHole은 결과로 나온 polygon이 Hole인지 유무를 가지는 1Byte형 point 구조이며, 전체 객체수는 nPart라는 4Byte형 자료에 담겨 나온다.

Tiling 단위는 4Level(Approach chart)기준 2분 단위로 Tiling을 하였다. 2분단위일 경우 한국 부근에서 일반적인 해상도 기준 약 3Km정도의 면을 가지며, 사용용도나 단말기의 해상도에 따라 적절한 선택이 필요할 것으로 판단된다.

Clipping 결과물중 Vertex Count가 4개인 객체들을 조사하여, Tile영역과 동일한 영역인 경우, Whole Data라고 판정하고, 이런 객체는 저장치 않고 특별한 Tag를 표시하여, Data Size를 줄이는 방법을 적용하여 실제 전체 Vertex Count를 줄이도록 하였다.

그리고, 큰 객체를 적은 영역으로 Clipping시, 많은 부하가 걸리는 부분이 있었는데, 이때는 Clipping 사이즈를 크게 잡아 2~3번 반복 하여 작은 영역으로 Clipping하는게 더 효율적인 방법이었다.

본 연구에서는 4Level을 2'단위로 Clipping하였어나, 이에 대해서는, 위에서도 언급한것 같이 좀 더 고찰이 필요한 것으로 사료되었다. 다른 1Level에서는 4도 단위를 사용하였는데, 한국 영역만이 아닌 전 세계를 기준으로 할때, 위도방향으로 적도선에서 나누어지지 않고, 적도 상하 2도씩을 가지는 Tile이 생성되었는데, 계산이 복잡해지는 경우가 발생하였다.

참고로 -180/-90을 원점으로 가지는 WorldWind 규격은 아래 표1과 같이 Level별로 2배수의 영역을 나누어 사용하는데, 적용을 고려해볼만한 규격인거 같다.

표 1. WorldWind Tile 규격
Table 1. Specification of WorldWind Tile

Lv	DEG	거리(Meters /512Pixel)
0	36.00000000000000	4000320.00000
1	18.00000000000000	2000160.00000
2	9.00000000000000	1000080.00000
3	4.50000000000000	500040.00000
4	2.25000000000000	250020.00000
5	1.12500000000000	125010.00000
6	0.56250000000000	62505.00000
7	0.28125000000000	31252.50000
8	0.14062500000000	15626.25000
9	0.07031250000000	7813.12500
10	0.03515625000000	3906.56250
11	0.01757812500000	1953.28125
12	0.00878906250000	976.64063
13	0.00439453125000	488.32031
14	0.00219726562500	244.16016
15	0.00109863281250	122.08008
16	0.00054931640625	61.04004

4. 프로토타입 구현 및 결과고찰

그림 8의 왼쪽은 한국해양조사협회 간행 Approach Chart 82종 중 육상영역정보만을 추출하여, 하나의 shape file로 merge한 data를 표시한 것이고, 오른쪽은 Tiling Tool을 이용하여 2분단위로 Tiling한 결과에 대하여 부산지역을 예시로 확대하여 보여주고 있다.

왼쪽 Data는 GlobalMapper를 이용하여 Shape File을 표시한것이며, 최종적으로 Tiling된 Data는 별도의 Viewer를 제작하여 표시한 화면이다.

별도의 Viewer는 해당 화면의 중점과, Scale, 화면 크기등의 정보를 이용하여, 필요한 Level및 Tile을 계산하고, Index Map을 이용하여, 필요한 Tile Data만 Load하여 Display하도록 만들어 졌다. 이때 Load된 Tile Data는 별도의 Cache관리를 하여, 지속적으로 여러번 Load되는 Tile은 Cache에 관리하고, Load된 cycle을 관리하여, 오래된 Cache순으로 버리는 방식을 취하여, Load되는 Data량의 최소화 및 적정량의 메모리 사용량을 보장하도록 하였다.

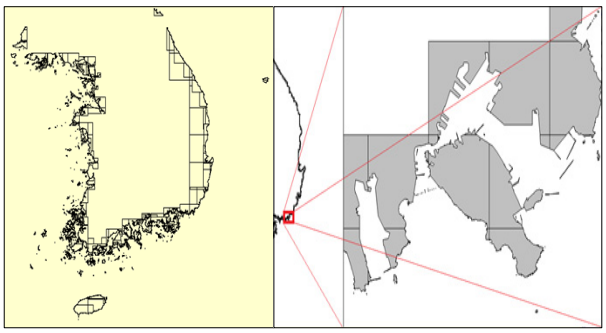

그림 8. Approach 차트 및 2'단위 타일링
Fig. 8 Approach Chart & Tiling Data Per 2sec.

표 2. Tiled-SENC 변환결과
Table 2. Result of applying Tiled-SENC

구분	S-57 ENC	Tiled-SENC
File/Tile 수	82	2,513
Area Count	11,019	14,823
Vertex Count	1,467,056	1,431,363

표2에서 보면 기존의 데이터를 Tiled-SENC로 변환한 경우 객체수는 증가하지만, 변환중 개별의 Tile영역과 동일한 Whole Data를 제거하는 방법을 적용하여 전체 Vertex Count는 줄었다는 것을 확인할 수 있었다.

또한 기존 S-57 ENC가 82 cell로 구성된 반면, Tiled된 경우는 2513개의 Tile로 구성이 된다. 결과적으로 비교영역에서 실제 처리해야 할 data량을 cell/tile수로 산정해보면, 2513/82=30으로, Tiled된 경우 약 30배의 처리 data 감소 효과가 있는 것을 확인할 수 있었다.

표2는 2'단위의 4Level기준이지만, 다른 Level에서도 유사한 결과를 도출할 수 있었다. 따라서, 원본 Data의 왜곡없이 저사양 단말에서의 성능보장을 확인할 수 있었다.

표 3. Level별 Vertex, Part 최고값
Table 3. Max. Count of Vertices and Part per Level

Level	Max. Vert	Max. Part
KR1	13,778	778
KR2	136,783	656
KR3	87,392	553
KR4	98,960	436
KR5	16,574	234

표3은 각 Level별 객체를 Full-Topology화 한 경우, 최대 Node및 Part를 표시한것이다. KR2에서 하나의 객체가 가지는 Node가 136,783으로 최고치를 보였다. 실제 KR2의 Chart수가 13장이고, 적정 Scale이 1/350,000~1/449,999인점을 감안한다면, 단말에 표시하기 위한 점의 개수가 너무 많다는 점이다. 그리고, KR1에서 최대 Part수가 778개인데, 저사양 단말에서 polypolygon을 처리하기위한 부담이 크다는 점을 고려한다면, 추후 개선할 부분으로 보여진다. Level별 적당한 polygon simplify 및 Hole객체의 분리 표현 방안등을 적용한다면, 단말기 화면 표시상 왜곡없이 더욱 효과적인 성능을 보장할 것으로 보여진다.

참 고 문 헌

[1] 심우성, 박재민, 서상현, "갱신을 고려한 전자해도 소형화 연구," *한국항해항만학회지*, 27권 4호, pp. 425-430, 2003.

[2] 오세웅, 박종민, 이문진, 고현주, "전자해도의 KML 변환기술 개발," *한국항해항만학회지*, 35권 1호, pp. 9-15, 2011.

[3] Joseph O'Rourke, "Computational Geometry," *Cambridge University Press*.

[4] 이희용, "S-57공간정보 저장을 위한 효율적인 SENC 구조의 설계 및 구현," *한국항해항만학회지*, 28권 제8호, pp. 673-678. 2004.

[5] ESRI Shapefile Technical Description, An ESRI White paper-July 1998

[6] Vatti, B.R. "A generic solution to polygon clipping," *ACM Commun.*, 1992.

저 자 소 개

조기정(Gi-Jong Jo)
1988년 : 한국해양대학교 항해학과 공학사
2011년~현재 : 한국해양대학교 대학원 해상교통학과 석사과정
2010년~현재 : GMT R&BD 센터장

관심분야 : Fuzzy, Recognition, Soft Computing
Phone : 02-488-6502
E-mail : jgj@gmtc.kr

이주환(Ju-Hwan Lee)

2005.2 호남대 전자공학과 학사
2007.8 한양대 컴퓨터공학 석사
2010. 8 한세대 정보보호학 박사
2003~현재 GMT 기술연구소 CTO

관심분야 : Embedded System, Soft Computing
Phone : 02-488-6501
E-mail : jlee@gmtc.kr