DirectX에 의한 지형 모델 뷰어의 구현

손 광 현[†] · 노 용 덕[††]

요 약

Direct3D는 새로운 3차원 그래픽 가속기로써, 이를 이용하여 보다 쉽게 윈도우용 그래픽 프로그램을 제작할 수가 있다. 여기서는 Direct3D 를 이용하여 제작한 프랙탈 지형 데이터 모델 뷰어를 제시한다. 지형 모델 뷰어 제작에 따른 클래스의 정의, 입력을 위한 대화상자의 설계, 초기화 과정, 프로그램의 실행과정을 설명하고 지형 모델 뷰어의 실행으로 얻어진 결과중, 철사형, 솔리드형, 포인트형의 출력를 보인다. 지형 형상화의 결과를 보이기 위하여, 여기서는 프랙탈 격자형 중간점 이동 알고리즘에 의하여 만들어진 데이터를 사용하였다.

Implementation of Terrain Model Viewer by DirectX

Kwang Hyun Sohn[†] · Yong Deok Noh[††]

ABSTRACT

Direct3D is a recently developed 3D graphic accelerator such that we could make a Windows graphic programs with Direct3D easily. This paper presents the fractal terrain model viewer developed in terms of Dierect3D. The object classes and the input dialog box used the model viewer, program initializing process, and the flow of the model viewer are discussed. Finally, the output of terrain formulation in wire-frame, solid, and point type form are presented. To show the results, the data of a terrain model were made by a fractal technique, the midpoint displacement methods with square lattices of points.

1. 서 론

멀티미디어, 게임, 및 가상현실등의 분야가 발전함에 따라서, 이들 분야에서의 사실적인 그래픽에 대한 요구가 점점 더 커지고 있다. 이러한 추세에 따라서 고가의 그래픽 웍스테이션에서만 가능했던 3차원 그래픽 가속 기능들이 가격이 저렴한 PC용으로 개발 발표되고 있다. 이런 시점에서, 마이크로 소프트에서는 Windows95의 멀티미디어 기능 및 게임 프랫폼으로써의 기능을 확장하기 위해서 그들의 OLE 기술을 기반으로 하는 DirectX SDK(Software Development Kit)을 개발하였다. 이 DirectX 기술은 각종 하드웨어 장비에 대하여 Windows에서 최소한의 추상 계층을 제공함으로써 계층적 운영체제의 단점인 낮은 효율을 극복하기 위해 개발된 것으로, 멀티미디어 구성에 필요한 그래픽 카드, 사운드 카드, 네트워크 카드, 조이스틱, 등에 대한 세부적인 사항을 포함하고 있다[9]. 또한 최근에 발표한 DirectX 에 3차원 가속기에 관한 표준인 Direct3D가 추가됨으로써 3차원 그래픽 가속기를 이용한 사실적이면서도 높은 프레임 기능을 갖는 Windows 프로그램 제작이 가능하게 되었다[5]. 여기서는 DirectX를 이용하여 제작한 프랙탈 지형 모델 뷰어 (이하 LandV)에 대하여 논하고자 한다.

† 정 회 원:한격레 정보통신 연구원
†† 종신회원:세종대학교 전산과학과 부교수
논문접수:1997년 4월 9일, 심사완료:1997년 8월 13일

지형 모델 뷰어에서 사용할 지형 데이터는 실제 데이터를 사용하기 보다는 프랙탈 기법에 의하여 만들어진 3차원 데이터를 사용한다. 이는 프랙탈 기법을 사용함으로써 지형의 지형의 형태를 갖는 비규칙적인 데이터 생성이 용이하며[2, 3, 4, 7, 8], 그 결과도 매우 바람직하기 때문이다. 비규칙적인 형태를 갖는 지형을 모델화하기 위하여 여러가지 프랙탈 기법이 만들어졌는데, 일반적으로 크게 나누어 다음과 같은 다섯가지 방식이 있다[11]. 즉, (1)Poisson faulting, (2) Fourier filtering, (3)Midpoint displacement, (4)Successive random addition, (5)Summing band-limited noises 등이 바로 그것으로, 각각의 방식마다 특색있는 지형 데이터가 생성된다[6, 10, 13, 14]. 여기서는 세번째 방식에 의하여 지형 데이터를 생성한다. 이 방법은 일반적으로 삼각형을 기준으로 지형 데이터를 생성하나, 여기서는 사각형을 기준으로 데이터를 생성하는 격자형 중간점 이동 알고리즘 (midpoint displacement algorithmwith square lattices of points)[12]에 의하여 지형 데이터를 생성한다. 프랙탈 기법에 의하여 임의의 지형 데이터를 생성하더라도, 이를 형상화하거나 애니메이션 효과를 얻기 위하여는 별도의 프로그램을 개발하거나, 기존의 그래픽 프로그램을 이용하여야 한다. 별도의 프로그램을 새로이 작성하기 위하여는 형상화에 필요한 이론적 배경 및 프로그래밍이 필요하며, 기존의 프로그램을 사용하기 위하여는 생성된 지형 데이터를 프로그램에 맞게 새로이 정리하는 작업이 필요하다. DirectX와 Direct3D의 사용은 위와 같은 작업 부담을 덜 수 있고 코드 재사용, 애니메이션, 실시간 랜더링, 등의 장점이 있다. 따라서, 이 논문에서는 여기서 사용할 DirectX와 Direct3D에 대하여 개괄적으로 살펴보고, 지형 모델 뷰어 제작에 따른 클래스에 대한 정의, 초기화 과정, 클래스에 의한 프랙탈 지형의 생성과 지형 모델 뷰어의 실행과정 및 지형 모델 뷰어의 실행에 따른 여러 가지 출력을 보이고, 마지막으로 지형 모델 뷰어 제작에 따르는 문제점를 논한다.

2. DirectX와 Direct3D

2.1. DirectX

DirectX는 MS-Windows 하에서 멀티미디어 및 게임 프로그램을 개발하고자 하는 프로그래머들을 위한 표준화된 개발환경을 제공한다. 또한 DirectX의 최신 버전에는 PC용 3차원 그래픽 가속기인 Direct3D가 새로이 추가되었다. 멀티미디어 개발환경을 제공하기 위해 만들어진 DirectX는 기존의 Windows API들과 다른 특징을 갖고 있다[9]. 첫째로, 기존의 컴퓨터 시스템의 자원을 고성능의 실시간 처리를 위해 응용프로그램에서 하드웨어에 대해 직접 접근이 가능하도록 개발되었다. 즉, 운영체제의 계층성을 파괴하지 않으면서 하드웨어에 대한 접근 계층을 최소한으로 줄이는 방식을 선택하였다. 따라서, 응용프로그램과 하드웨어 사이에 있는 중간 처리 과정의 생략으로 실시간 처리가 용이하고 하드웨어 가속 기능을 효율적으로 사용할 수 있는 조건이 형성되었다. 둘째로, 하드웨어 제작자와 사용자 그리고 개발자에게 일관된 인터페이스를 유지하도록 하였다. 이는 공통된 표준 마련을 뜻하는 것으로, 개발단계부터 하드웨어 제작사들과의 연계를 통해서 DirectX를 새로운 표준으로 자리잡게끔 하고있다. 이로써 그 동안 실시간 대화형 소프트웨어 개발을 위해 개발자에게 지워져 있던 하드웨어마다의 특성에 맞는 모듈을 모두 개발하여야 부담이 사라지게 되면서 표준화된 개발환경 구현이 가능하게 되었다.

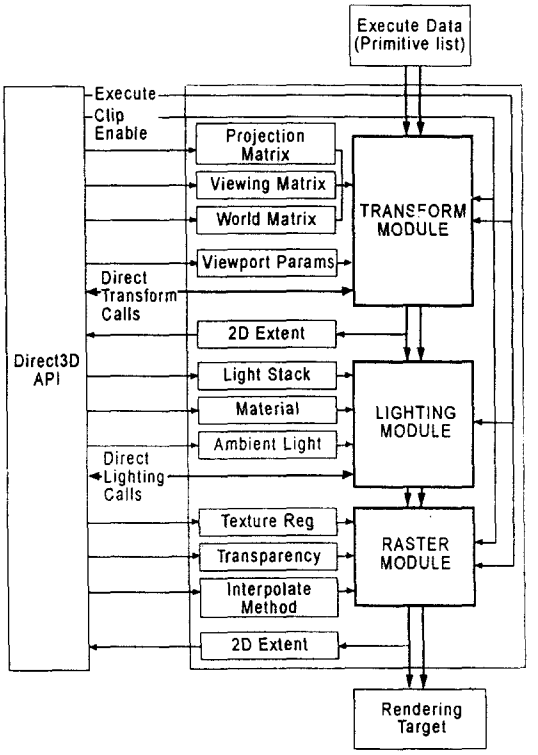
실시간의 대화형 멀티미디어 및 게임 소프트웨어 개발을 위해 개발된 DirectX의 구성 요소에는 개인용 컴퓨터에서 사용하는 모든 장비에 대한 처리기술이 포함된다. 여기에 해당하는 장비로는 GUI 환경에서의 2차원 그래픽 가속기, 사운드 카드, 네트워크 카드, 모뎀, TV 카드, 및 3차원 그래픽 가속기 등을 꼽을 수 있다. 이들 장비에 대한 DirectX의 각 구성요소들은 OLE에서 사용하는 COM(Component Object Model)으로 구성되어 있다.

COM은 코드 재사용에 초점을 둔 객체지향시스템의 기초이며, OLE 프로그래밍의 핵심이 되는 모델로 다음과 같은 특징이 있다. 첫째, COM은 객체에 대한 새로운 인터페이스의 추가가 용이하다. 둘째, COM은 실행파일이나 혹은 DLL, 심지어 네트워크상의 다른 컴퓨터에서도 실행 될 수 있다. 이것은 OLE의 중요한 기능 중의 하나로, 필요한 객체의 위치만 안다면 어디서든 실행시켜 그 기능을 사용할 수 있는 편리성을 제공한다. 셋째, COM은 프로그램이 수행되

는 중간에 로드되어 사용되다가 더 이상 사용되지 않으면 자동으로 메모리에서 사라진다. 이것은 사용자가 특별한 처리를 하지 않아도 자동으로 사용되지 않는 객체들을 메모리에서 제거하여 공간을 절약하는 효과를 나타낸다[1, 9].

2.2. Direct3D

Direct3D는 실시간의 3차원 그래픽에 관한 모든 기능과 자원을 제공하는 DirectX의 구성요소이다. Direct3D는 세 가지로 구분된 모듈에 의해 구성되는 가상의 3차원 랜더링 엔진(rendering engine)에 그 기초를 두고 있으며, 이 세 가지 모듈은 수행 버퍼(execute buffer)에 의해서 제어된다. 이 수행 버퍼는 Direct3D 랜더링 엔진에 대하여 처리해야 할 3차원 데이터와 그 데이터에 대한 처리 명령어들을 갖고 있는 하나의 리스트이다. 따라서 랜더링 엔진은 이 리스트를 처리함으로써 정지되거나 혹은 애니메이션되는 3차원 이미지를 만들어낸다.



(그림 1) Direct3D 랜더링 엔진의 구조.
(Fig. 1) The structure of Direct3D rendering engine.

랜더링 엔진은 전환작업모듈 (Transformation Module), 광선처리 모듈 (Lighting Module), 화면출력 모듈 (Rasterization Module)로 구성되어 있다 (그림 1). 이 세 가지 모듈들은 각각의 처리를 위한 상태정보를 유지하면서 엔진에 대하여 입력으로 들어오는 수행버퍼를 처리한다. 또한 이 모듈들은 각각 필요할 때마다 로드되어 사용되기 때문에, 언제든지 하드웨어의 가속기능과 다양한 랜더링 효과를 내기 위해 교체되어 사용될 수 있을 뿐만 아니라, 각각의 모듈들에서 하드웨어가 지원하지 않는 기능들을 에뮬레이션을 하는 기능도 제공한다. 수행버퍼는 랜더링 엔진에서 처리해야 할 모든 정보, 즉, 도형의 꼭지점 리스트, 선과 평면에 관 한 정보, 표면의 질감, 변환에 관련된 행렬의 정보 등을 담고 있는 패킷과 같은 것으로, Direct3D에서 사용하는 하나의 자료구조이다.

3. 지형 데이터 뷰어의 제작

Direct3D의 랜더링 엔진을 이용해 지형에 대한 3차원 이미지를 생성하고자 할 경우에, 먼저 Direct3D의 각 module들에 대한 초기화 과정을 거쳐야 한다. 이 때, 수행버퍼를 구성하기 위하여 생성하고자 하는 3차원 데이터와 그 처리를 Direct3D에서 정의한 데이터 형식에 맞게 나열하여야 한다[5, 9]. 이를 위하여 지형 데이터 생성에 필요한 클래스를 정의하였다.

3.1. 클래스의 정의

지형 데이터 뷰어(LandV)에서는 Direct3D의 3차원 그래픽 기능을 최대한 활용하고, 지형 데이터 생성 부분의 모듈화와 코드의 재사용성을 높이기 위해서 객체지향 기법을 적용하여 C++ 클래스로 제작하였다. 이를 위해, LandV는 격자형 중간점 이동 알고리즘 구현에 관한 모든 정보와 함수를 포함하고 있는 객체 CMidPoint 클래스와 생성된 데이터를 저장하는 2차원 배열 클래스 CLandScape를 정의한다.

CLandScape 클래스는 이차원 배열의 생성과 파괴, 그리고 지형 데이터 결과의 저장 및 참조를 용이하도록 하는 지형 데이터 저장객체이다. 이 클래스의 제작 목적은 단순하고 복잡한 작업을 자동화시키데 있다. 따라서 그 작업의 대부분은 사용자로부터 은폐되

```
// Horizontal Data Line
class C_X {
        float *x ;
        UINT   xSize ;
public :
        C_X () {x = NULL ;};
        ~C_X() {if (x) delete x ;};
        float & operator[] (int index) {return x[index];}
        C_X   & operator=  (C_X &obj)
                           {memcpy (x, obj.x, sizeof(float)*obj.xSize);
                            return *this;}
        LS_RETURN FInit (UINT N)
        {
                if (N <= 0) return LSERR_FAIL ;
                if ( x != NULL) delete x ;
                if ((x = new float[N]) == NULL) return LSERR_OUTOFMEM ;
                memset(x, 0, sizeof(float)*N) ;
                xSize = N ;
                return LS_OK ;
        }
} ;
// Vertical Data Line
class CLandScape {
        C_X  *y ;
        UINT  ySize ;
public :
        CLandScape (const CLandScape &obj) ;
        CLandScape () {y = NULL ;} ;
        ~CLandScape() {if (y) delete []y;}
        LS_RETURN FInit (UINT N) ;
        UINT      GetSize() const {return ySize;} ;
        C_X        & operator[] (int index) {return y[index] ;}
        CLandScape & operator=  (const CLandScape &obj)
        {       if (FInit (obj.ySize-1) == LS_OK)
                for (UINT i = 0 ; i < obj.ySize; i++) y[i] = obj.y[i] ;
                return *this;
        } ;
} ;
```

(그림 2) LandV의 지형 데이터 저장 2 차원 배열 클래스의 정의.

(Fig. 2) The class definition for 2D terrain data array.

어 있고, 작업수행을 위한 일부 함수들만을 공개하고 있다 (그림 2). 공개된 함수는 초기화 함수 FInit를 비롯하여 배열의 크기를 알아내는 GetSize 함수와 두 개의 연산자 Overload 함수가 있다.

CMidPoint 클래스 설계의 기본 목표는 클래스 내부에서 생성되는 데이터와 그 생성 과정에 대한 정보를 공개하지 않음으로써 코딩 중에 발생할 수 있는 오류를 최소화하여 프로그램의 안정성을 확보하는 데 있다. 이런 구조적 특징으로 인해 CMidPoint는 지형 생성을 위한 객체 인터페이스를 제외한 모든 정보들은 비공개로 선언하고 있다(그림 3). 따라서 이 클래스를 이용하여 데이터를 생성하기 위해서는 CMidPoint 클래스에서 제공하는 FInit, GenLandScape, GetLandData의 3가지 공개함수로 구성된 인터페이스를

```
class CMidPoint {
private:
        CRand       gauss ;
        CLandScape X ;
        int         maxLevel ;
        int         N ;
        float       delta ;
protected :
        float      F3 (float x0, float x1, float x2) ;
        float      F4 (float x0, float x1, float x2, float x3) ;
public :
        BOOL FInit (int maxLevel, UINT seed);
        BOOL GenLandScape(float sigma, float H, BOOL addition) ;
        BOOL GetLandData (CLandScape &obj, INT scale) const;
} ;
```

(그림 3) LandV 의 지형 생성 클래스의 정의.
(Fig. 3) The definition of a terrain data creation class.

이용한다.

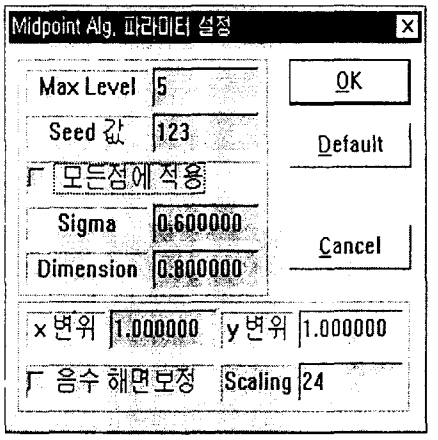
먼저, FInit 함수를 호출하여 격자형 중간점 이동 알고리즘에서 사용하는 정규분포 난수 생성자의 초기값과 반복 회수를 초기화한다. 또한 반복회수에 의해 결정되는 지형의 크기에 따라 CLandScape를 초기화한다. 다음에는 GenLandScape 함수를 호출하여 데이터를 생성한다. GenLandScape 함수는 지형 데이터를 생성하기 위해 격자형 중간점 이동 알고리즘을 구현한다. 즉, 이 함수는 지형의 복잡도, 생성을 시작하는 초기 사각형의 한 변의 길이, 생성의 각 단계에서 모든 점에 대해 난수값을 더할 것인가를 결정하는 부울 인자를 받는다. 여기서 생성된 데이터는 FInit함수에서 초기화한 CLandScape 객체에 저장된다. 마지막 작업은 생성된 데이터의 복사본을 받아내는 단계로 GetLandData 함수는 인자로 받는 스케일 값을 이용하여 불규칙한 데이터를 재스케일링하고 그 결과를 CLandScape의 새로운 Instance에 저장해 되돌린다.

3.2. Direct3D의 초기화

Direct3D의 초기화는 프로그램에서 Direct3D 객체를 사용하기 위해서 거쳐야할 단계로 1)시작준비단계, 2)Direct3D COM 생성, 3)Viewport 초기화, 4)랜더링 초기화 과정으로 요약될 수 있다. 시작준비단계에서는 Direct3D를 실행할 Windows를 생성하고 그에 대한 초기화 즉, 윈도우 실행에 필요한 메뉴 및 자원들을 로드하게 된다. Direct3D를 실행할 Windows에 관한 모든 초기화 작업이 끝난 후에는, Direct3D COM을 생성하게 된다. 세 번째 Viewport 초기화 단계에서는 생성하고자 하는 viewport에 대해 배경 및 빛을 설정하는 과정으로 랜더링에 영향을 미치는 빛에 대해 설정하게 된다. Direct3D의 마지막 단계는 Direct3D 랜더링 엔진의 세 모듈들에 대해서 각각의 초기상태를 설정하는 과정이다. 이 과정을 통해 랜더링이 시작되는 시점의 카메라의 위치, 객체의 초기상태 등을 설정할 수 있다. 이러한 초기화 과정을 거치고 나서 랜더링 엔진의 수행화일에 수행버퍼를 제공하면, 랜더링된 이미지를 화면에서 볼 수 있는 상태로 Direct3D의 실행 준비가 끝나게 된다.

3.3. 프랙탈 지형의 생성과 LandV의 실행과정

격자형 중간점 이동 알고리즘을 프랙탈 지형 데이터를 생성하기 위해서는 난수의 초기 값과 같은 몇 가지 기초 값이 필요하다. 따라서 격자형 중간점 이동 알고리즘을 이용하여 지형 데이터를 생성해내는 CMidPoint::GenLandScape (그림 1 참조)에 이러한 인자를 넘겨주어야 하는데, 지형 모델 뷰어(LandV)에서는 이 초기 값을 사용자로부터 용이하게 받아들이기 위해서 (그림 4)과 같은 윈도우용 대화상자를 이용한다. (그림 4)에서 보는 바와 같이 대화상자의 입력항목은 모두 기본 값을 갖고 있으며, 사용자의 필요에 따라 손쉽게 초기 값을 변화시킬 수 있는 인터페이스를 제공한다.

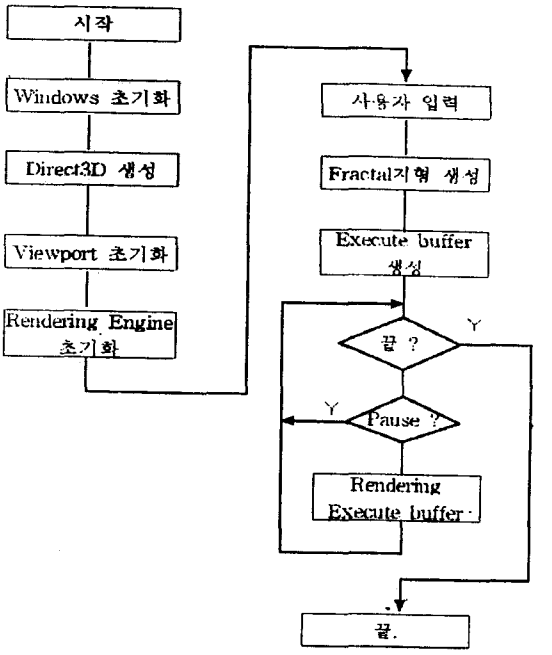


(그림 4) 초기값 입력 대화상자.
(Fig. 4) Input dialog box.

LandV는 사용자로부터 받아들인 초기 값을 이용해 CMidPoint 클래스로 부터 격자형 프랙탈 지형 데이터를 생성해 낸다. 하지만 이 알고리즘이 사각형의 격자모양 데이터를 생성해내는데 반해 Direct3D는 기본적으로 삼각형의 다각형만을 지원하기 때문에, CMidPoint::GetLandData로 받아낸 데이터를 변환시키는 과정이 필요하다. 이를 위해서 LandV는 여러 개의 삼각형을 결합하여 하나의 다각형으로 생성하는 기법을 사용한다. 즉, 한 사각형 당 두 개의 삼각형 형태의 다각형으로 분해한 뒤, 다시 하나로 결합하는 방법으로 수행버퍼에서 평면 데이터를 구성한다. 그리고, 3차원 데이터외에 지형의 중앙을 중심으로한 360° 회전 애니메이션을 위해서 전환작업모듈에서 관리하는 전체좌표 변환행렬에다가 이동작업, 회전작업, 이동작업의 행렬을 곱하는 명령을 추가시킴으로써 수행버퍼의 구성이 끝나게 된다. 수행버퍼가 생성되고 나면 주 프로그램의 랜더링 루프[9] 구조를 통해 랜더링 엔진에 반복적으로 입력되어 화면을 구성하게 된다. 결과적으로 랜더링 루프에 의하여 수행버퍼를 실행할 때마다 모든 다각형에 대하여 버퍼에 정의되어 있는 회전행렬이 적용되기 때문에, LandV에서의 지형은 회전하는 것처럼 보여지는 결과가 나온다.

지금까지 설명한 LandV의 실행 절차를 요약하여 순서도로 나타내면 (그림 5)와 같다.

이외에도 LandV는 키보드를 이용한 카메라의 조작으로 사용자 임의대로 지형에 대한 관찰 위치를 찾을 수 있는 기능을 포함하고 있다. 이 키보드에 의한 카메라의 위치변화는 주 프로그램의 랜더링 루프에 의해서 수행버퍼가 실행됨으로써 화면에 나타나게 되는데, 스페이스 바에 의해서 랜더링 루프가 정지한 상태에서는 화면이 수정되어 출력되지 않으므로 변화가 나타나지 않는다. LandV에서 제공하는 각 키에 대한 카메라의 위치 변화를 정리하면 〈표 1〉과 같다.



(그림 5) LandV 의 실행 흐름도.
(Fig. 5) The flowchart of LandV.

〈표 1〉 키보드를 이용한 카메라의 위치 이동
〈Table 1〉 Translation of camera position by keyboard.

		+Shift	+Ctrl
Up	X축 회전 +	Y축 +이동	Z축 +이동
Down	X축 회전 −	Y축 −이동	Z축 −이동
Left	Y축 회전 +	X축 −이동	Z축 −회전
Right	Y축 회전 −	X축 +이동	Z축 +회전

4. LandV에 의한 지형 형상화 작업

앞에서 설명한 클래스들을 기본으로 만들어진 프

랙탈 지형 모델 뷰어 LandV를 사용하여 지형 형상화 작업을 수행하였다. 먼저 격자형 중간점 이동 알고리즘을 이용하여 지형 데이터를 생성하여야 하는데, 이를 위하여 (그림 4)의 대화상자를 이용하여 필요한 값을 입력한다.

(그림 4)에서의 Max Level은 중간점을 구해나가는 단계의 횟수를 나타낸다. 위의 예에서 5의 값의 경우는 한 변에 2**5=32개의 점이 생기고 결과적으로 모두 1024개의 점이 사각형내에 만들어진다. Seed는 지형 데이터 생성에 필요한 난수의 초기 값이며, Sigma는 사각형의 한 변의 길이를 나타낸다. X 변위와 Y 변위는 사각형 분할시에 분할된 사각형의 한 변의 기본길이이며, 이 값을 조정함으로써 뷰어내에 지형의 모습을 적당한 크기로 출력할 수가 있다. 또한, Scaling은 지형의 높이를 조정하기 위한 값으로, 값이 커질수록 평평한 지형이 만들어진다. 음수해면보정은 지형의 높이를 구하였을 때, 음수의 값은 모두 0의 값으로 지정하는 것을 의미한다.

마지막으로 (그림 6) 및 (그림 7)은 LandV를 사용하여 만들어진 결과이다. (그림 6)은 음수해면보정을 하지 않아서 지표면이 전부 다 울퉁불퉁한 모습을 보여주고 있다. (그림 6-a)는 지형을 철망형으로 출력한 경우이고, (그림 6-b)는 솔리드 형태로, (그림 6-c)는 점의 형태로 출력한 경우이다. 어떤 형태로 출력할지는 뷰어의 메뉴중 파일 메뉴에서 선택하면 된다. 이러한 출력은 360도 회전하는 동영상의 모습으로도 볼 수 있으며, 정지된 형태로도 볼 수 있다. 또한, 정지된 형태에서 단계적으로 움직이게 할 수도 있다. 그리고, **〈표 1〉의 키를 사용하여 카메라 위치를 변동**시킴으로써, 가까이, 혹은 멀리 보이게 하는 효과는 물론 X, Y, Z 축을 기준으로 360도 회전시켜서 특정 위치에서 지형 형상화의 결과를 살펴 볼 수도 있다.

(그림 7)은 (그림 6)과 같은 지형으로 단지 음수해면보정을 한 결과이다. 음수해면보정은 지표면의 높이가 음수인 경우에는 모두 해면으로 간주한다. 결과적으로 지형의 모습이 바다위의 섬의 모양, 또는 해변가의 모습을 보여주고 있다.

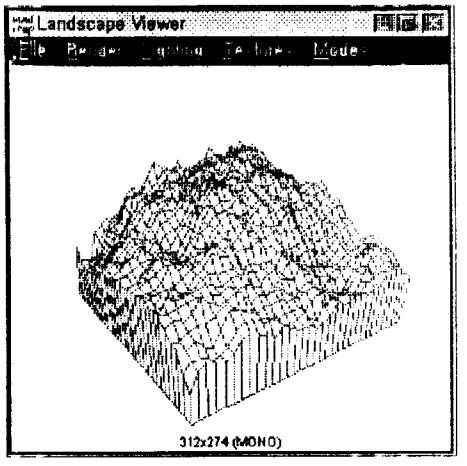


그림 6-a

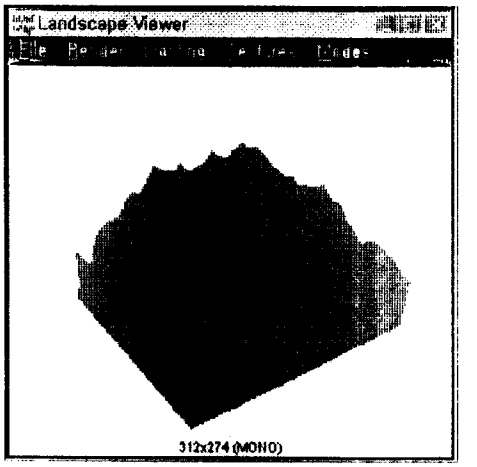


그림 6-b

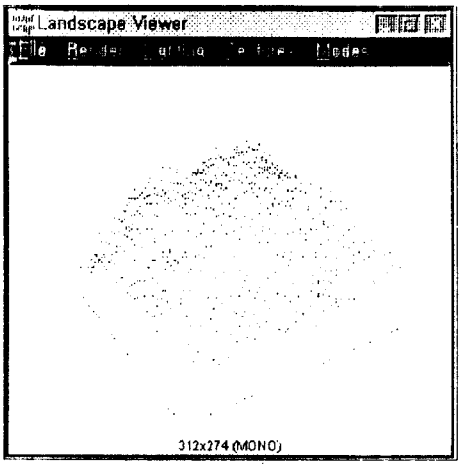


그림 6-c

(그림 6) 음수해면보정을 하지 않은 지형.
(Fig. 6) The terrain surface with a negative height.

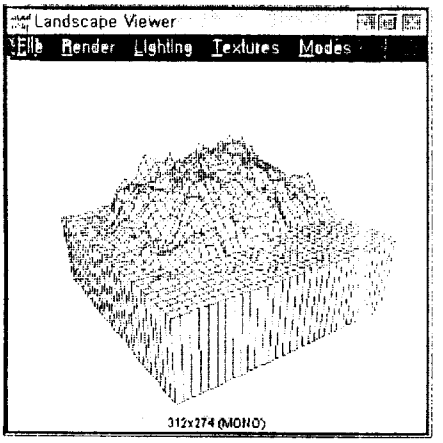


그림 7-a

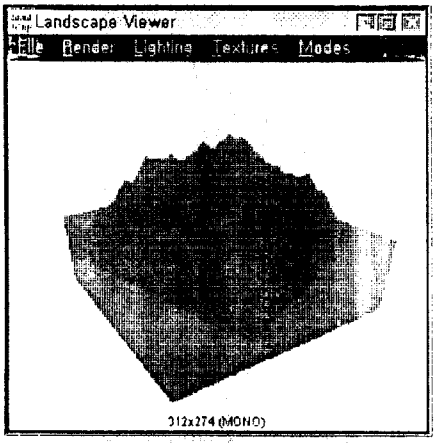


그림 7-b

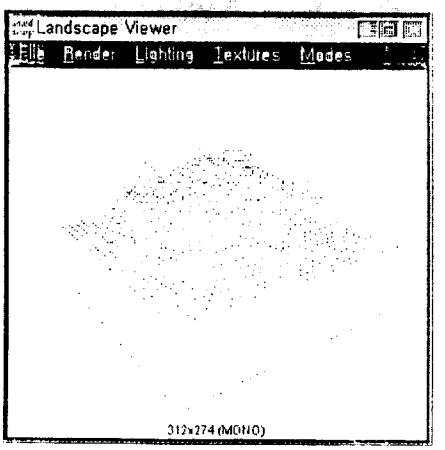


그림 7-c

(그림 7) 음수해면보정을 한 지형.

(Fig. 7) The terrain surface without a negative height.

5. 요약 및 결론

MS-Windows 하에서 응용프로그램을 제작하는 사람들에게 표준화된 개발환경을 제공하고자 했던 DirectX가 객체지향 개발기법에 있어서 새로운 기술 혁신을 가져온 OLE를 이용해 운영체제 자체에 구축되면서 새로운 형태의 Windows 개발환경이 마련되었다. 특히, DirectX에 추가된 Direct3D는 지금까지의 3차원 그래픽 프로그램 개발자들에게 주요 과제로 남아있던 하드웨어 가속기능 이용에 관한 표준을 마련했다는 점에서 의미를 갖고 있다. 여기서는 DirectX의 기능을 일부 이용하면서 객체지향 개발기법에 따라 지형 형상화에 필요한 클래스의 일부분에 대하여 보이고, 그 실행순서와 여러 가지 가능한 결과를 보였다.

비록, 여기서 DirectX 내의 Direct3D의 기능을 활용하기는 해도, 여전히 Direct3D라는 환경의 제약을 받는다. 실제로, 데이터의 생성에 필요한 클래스를 정의하고, 입력을 받고, Direct3D를 제대로 활용하기 위한 데이터 처리과정을 위하여 많은 작업이 소요되었으며, LandV의 경우는 1000 라인에 가까운 프로그래밍 작업을 거쳐서 만들어진 것이다. 이러한 문제점은 그래픽이나 멀티미디어 작업에 편리한 표준화된 개발환경이 점점 더 발전함에 따라서 점차로 해소될 것으로 기대한다.

참 고 문 헌

[1] K. Braockschmidt, "Inside OLE 2", Microsoft press, 1994.

[2] A. Fournier, and W.T. Reeves, "A Simple Model of Ocean Waves", Computer Graphics, Vol. 20, No. 4, pp. 75-84

[3] A. Fournier, D. Fussell, and L. Carpenter, "Computer rendering of stochastic models", Communications of the ACM, Vol. 25, No. 6, pp. 371-84

[4] D. Hearn, "Computer Graphics", 2nd Ed., Prentice-Hall, 1987.

[5] Internet site, http://www. dimension3d. com

[6] J. P. Lewis, "Generalized Stochastic Subdivision", ACM Transactions on Graphics, Vol. 6,

No. 3, pp. 167-190, July, 1987.

[7] B.B. Mandelbrot, "Comment on Computer Rendering of Fractal Stochastic Models", Communications of the ACM, Vol. 25, No. 8, pp. 581-584, 1982.

[8] B.B. Mandelbrot, "The Fractal Geometry of Nature", W. H. Freeman, San Francisco, 1982.

[9] Microsoft, "DirectX 2 SDK Document", Microsoft Corp., 1995.

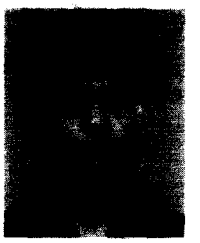
[10] Miller, and S. P. Gavin, "The Definition and Rendering of Terrain Maps", Computer Graphics, Vol. 20, No. 4, pp. 39-48, 1986.

[11] F.K. Musgrave, C.E. Kolb, and R.S. Mace, "The Synthesis and Rendering of Eroded Fractal Terrains", Computer Graphics, Vol 23, No. 3, pp. 41-50, July, 1989.

[12] H.O. Peitgen, and D. Saupe (eds.), "The Science of Fractal Images", Springer-Verlag, New York, 1988.

[13] K. Perlin, "An Image Synthesizer", Computer Graphics, Vol. 19, No. 3, pp. 287-296.

[14] R. Szeliski, and Demetri Terzopoulos, "From Splines to Fractals", Computer Graphics, Vol. 23, No. 3, pp. 51-60, July, 1989.



손 광 현

1997년 세종대학교 전산과학과 졸업(학사)

1997년~현재 한겨레 정보통신(주) 연구원

관심분야: 객체지향언어, 통신제어, 게임소프트웨어 개발

노 용 덕

1976년 서울대학교 산업공학과 졸업(학사)

1984년 Auburn Univ. I.E.(MS)

1987년 Auburn Univ. I.E.(Ph. D)

1976년~1980년 국방과학연구소

1988년~현재 세종대학교 전산과학과 부교수

관심분야: 컴퓨터 그래픽스, 시뮬레이션, 및 성능분석