TR-88-028

碎形自然物的動態生成法

TR-88-028

論文名稱：碎形自然物的動態生成法　　頁數：67

校(院)所組別：淡江　大學　資訊工程　研究所

畢業時間及提要別：七十七學年度第二學期碩士學位論文提要。

研究生：史國賢　　指導教授：鄭國揚　博士

論文提要內容：

自然界中的物體大部分具有複雜而不規則的形狀，像曲折的海岸線、山脈、雲的形狀及閃電的路徑，通常無法用歐氏幾何學來表現這些物體的自然特徵。因此Mendlebrot〔1977，1982〕創出碎形幾何學來表現物體的自然特性，這些特性包括：物體小部分和整體有相同結構的自我相似性，尖銳外形的不可微分性，拓樸外形的非整數維度以及物體各部分拓展及細分時形狀的一致性。

本論文參考Fournier et al.〔1982〕及Peitgon et al.〔1988〕等人的隨機中點替換法表現自然物體各個成長步驟的形狀，並記錄其成長過程，以模擬自然碎形物體的成長。在此方法中，使用者可控制一些擾亂的變數，使物體的外觀更自然而眞實。此方法的處理過程是使用者先輸入粗略的物體形狀，然後調整擾亂的程度以決定零碎的亂度，並依細分的程度控制形狀的變化量，當使用者決定擾亂變數的量，配合Gaussian分配的控制可產生隨機中點，使外形收斂於自然物體的形狀，並符合Mendlebrot所提碎形自然物體的特性。

Title of Thesis : Generating Natural Fratal Object in Animation

Total pages : 67

Name of Institute : Graduate Institute of Information Engineering, Tamkang University

Graduate Date : 06/ /89

Degree Conferred : Master

Name of Student : Kuo-Hsien Shih 史國賢

Advisor : Kuo-Young Cheng

Abstract :

Most objects in nature possess complex and irregular shapes, mountains, coastlines, cloud, and lightening are few examples among them. These objects in general can not present their natural properties in Euclidean geometry. Benoit B. Mendlebrot [1977,82] has devised fractal geometry to generate natural objects. The major concepts of fractal geometry is to explain some properties of natural objects. These properties include self-similarity, that is magnified subsets look like or identical to the whole, sharp fragments are non-differentiable, non-integer dimension of topological shapes, and consistency of extended and fractional shapes.

This thesis adopts random midpoint displacement algorithm proposed by Fournier et al.[1982] and Peitgen et al.[1988] to represent natural objects in growing. To simulate shapes in growing, the midpoint of each line segment are perturbated randomly. However, some perturbations are controllable by users. The user first input rough sketch of the object's shape, and then adjust the degree of perturbations to obtain the roughness of fractals and the variation of magnitude to obtain refinement according to sizes. We show that, by proper selection of parameters, the random midpoint contrlled by Guassian distribution will converge to the state with fractal properties mentioned by Mendlebrot.

目　錄

第一章　緒論

第一節　前言

在電腦繪圖(Computer Graphics)領域中，物體的表面模塑(Modeling)是最首要且基本的工作，一般而言模塑技術均假設物體是由一組線條或面所構成，當物體形狀非常簡單時，可輕易而精確地描述，例如三角錐為四個三角平面所組成，立方體由六個四邊形平面所組成。但要表示較複雜物體卻無法精確地描述物體形狀，須用Bezier, Coons, 或 B-spline等高次項式(High Degree Polynormial)來逼近(Approximation)，這些高次多項式能產生任意形狀的平滑(Smooth)表面，現今正被廣泛地應用在模塑剛性(Solid)物體，像雷諾車系的車體設計，建築物體架構的繪製均可由高次多項式表現。

儘管高次多項式能表示任意形狀的剛性物體，但對於真實世界(Real Word)中自然物體(Natural Object)的不規則(Irregular)外形卻很難描述，如曲折的海岸線、起伏的山脈等等；假設用Bezier來描述山脈的外形，則山脈的大致外觀也許很逼近，但細部看來Bezier多項式的平滑特性，便完全破壞了山脈的真實性；因為山脈的表面是由

許多不規則且紊亂(choas) 的面所組合而成，因此這種破碎(Fragement)且不規則(Irregular)的特性，無法用傳統的歐氏幾何模塑法(Eulidean Geometric Modeling)來表現，故孟德爾博(Mandelbrot B.B.)[1,2,3]在1968年提出碎形幾何學(Fractral Geometry)成為一門新的研究領域，他綜合了自然物體的碎形特性(Fractal Properties)並和傳統歐氏幾何特性作一比較。本論文將在第二章定義碎形物體(Fractal Objects)並說明碎形幾何與歐氏幾何之不同點。

鑑於碎形幾何模塑方法的廣泛適用性，本論文的目的在於注入這種模塑方法於中研院資訊所的電腦動畫系統(Computer Animation System)—動畫者(Animator)系統，補足動畫者系統在模塑功能的不足，讓使用者(User)輕易地輸入粗略的物體外形，便可得到複雜且細碎的物體形狀，以達到物體的真實效果，並提昇系統的易使用度(User-friendly)。

第二節　自然界的現象(Natural Phenomena)

這些零碎的特性早在百年前就被植物學家發現，而且陸續地被討論及鑽研，直到最近十幾年，由於電腦的長足

進步，纔揭開了自然物體零碎特性的神祕面紗，由於零碎的觀念涵蓋廣泛，不但在純數學(Pure Mathemetic)，及物理學(Physic)的領域中，成一獨立的分支，並可應用到化學(Chemistry)，地質學(Geology)及氣象學(Meteorology)等等，科學家們針對其需求，將自然界的各種物體的行為及現象，建立一預測模型，再利用零碎及機率(Properbility)的觀念，將預測模型建入電腦中模擬，以驗證模型的可靠性。以下列舉一些具有碎形特性的自然物：

・海岸線(Coastline)

海岸受到海浪及風化的侵蝕，甚至地震的影響，使海岸的輪廓曲折而零亂，就像許多細小線段拼湊而成的曲折線條。如圖1.1中零碎的海岸線實質上很難由高次曲線來逼近，就算能表現亦會花費超量的計算時間(Computation time)。同理圖1.2所示零亂電波亦是零碎的自然物體。

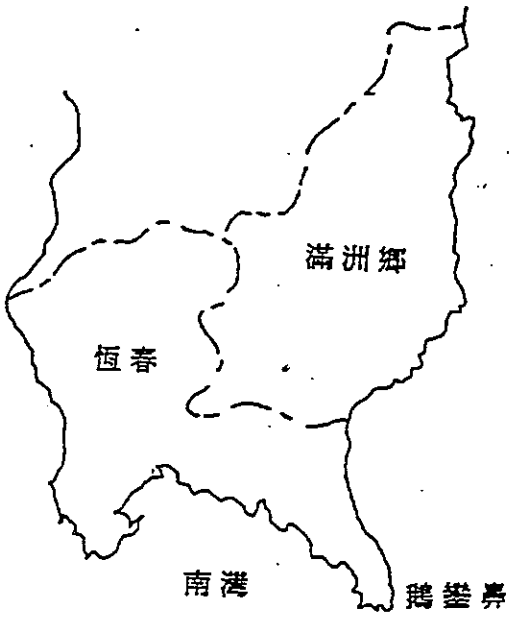


圖1.1 鵝鑾鼻附近的海岸線圖

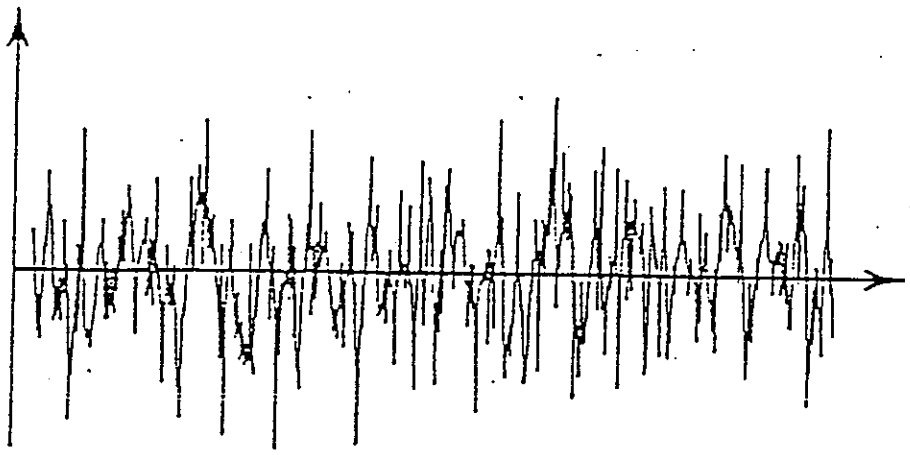


圖1.2 零亂的電波

・股票指數(The activity of stock market)

股票指數的漲跌就長期趨勢而言，有可預測的走向，而短期的變化卻難以預測，例如高科技工業的股票就長期投資應是上漲的趨勢，但可由投資者的投機心理，大戶的短期操縱或是人為的內線交易而使股價起落異常，形成零亂的升降曲線。如圖1.3是四個星期的短期股價走勢圖，漲跌變化相當亂。美國有一些投資人運用亂數產生器(Random Number Generator)來模擬股票指數的漲跌，結果這些投資人表示，模擬的預測比人為的預測準確。

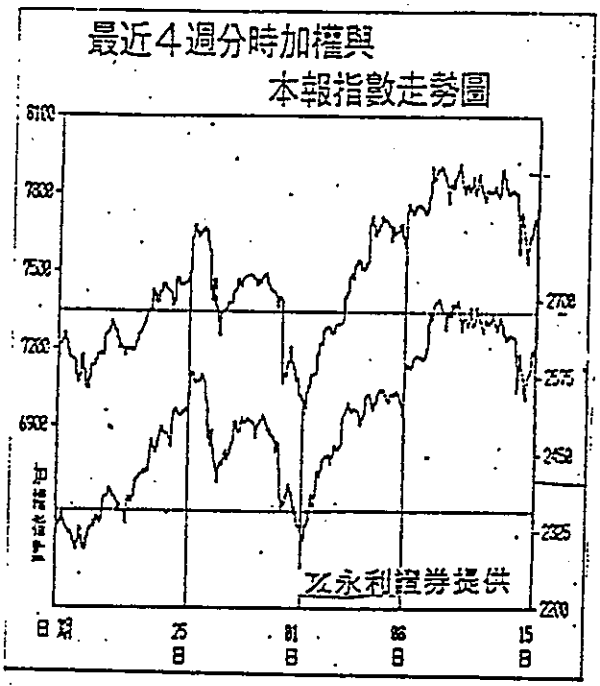


圖1.3 起起落落的股價指數

・山脈(Mountains)

由於地球大陸板塊運動及地殼變動影響，造山運動日夜不停，地質學家(Geologist)表示世界第一高峰聖母峰所在的喜瑪拉雅山至今仍受大陸塊板的推擠而繼續昇高。這些運動的模型必須仰賴專家來建立，依模型而能以電腦模擬，不但可以驗證模型的正確性，並可應用於各方面如飛行模擬，地震災害預報等等，不僅是科學界的發展，而且更對人類的安全多罩上一層保護。

・宇宙現象(Universal Phenomenon)

起伏無常的星球表面，也是零碎而不規則的，如同山脈的起伏一樣。擴而大之，宇宙間的物理模型(Physical Model)亦可由電腦來模擬其現象，如美國康乃爾大學依黑洞形成的物理現象，建立成一個物理模型，並用電腦模擬黑洞之運動過程，並推演而得到黑洞形成過程中的各種情況，以驗證說明黑洞這科技之謎。

・雲(Cloud)

美國國家氣象研究中心使用超級電腦CRAY X-MP 模擬雷雨雲的產生，依照大氣模型，取十萬點樣本點，計算各樣本點的風速，氣溫及濕度等，綜合各樣本點的結果，用以得知雷雨雲產生的氣象條件及產生過程中的溫、濕度變

化。同理，雨、霧等都和雲具有相同的特性。

第三節　電腦的模擬方式

由上述例子可知：自然碎形物體都是相當複雜且不規則的，碎形物體的處理過程都是重複地計算大量的數據，類似這種繁複的處理工作交給電腦是再適合也不過的。碎形幾何的觀念不但是各類自然科學的常用工具，同時也因碎形模塑能產生新穎且真實的物體外型，能提高圖片的真實效果，故碎形模塑的電腦繪圖能引起繪圖工作者及影片製作者的興趣，凡此種種都是碎形幾何學的魅力所在，尤其在電腦繪圖的生動柔化(Realistic Rendering) 及自然現象模塑部份扮演重要的角色，它透過繪圖廣泛地應用在科學界、甚至日常生活中的藝術(Art)、廣告(Advertis-ting)、飛行訓練(Flying Trainning)等等。

基本上，自然界的物體大部份由三度空間所呈現，但處理碎形物體的方法應視其對象之不同而有異；例如山是三維的物體，但處理的對象若是山的輪廓則為一維的線條，若處理山的表面則應歸類於二維的平面處理，同理若考慮山的整體體積則又是三維的處理方式，故依處理對象元件的不同分為以下數種：

(1)一維碎形法(One-dimensional Fractal)

顧名思義，此法處理對象為一維的線段(Polyline)。經由碎形化(Fractalize)可得一組細小線段組合的線條(Line)，且此線條形狀大體上和原線段(Polyline)相似，但細部的變化卻不大相同[4]。如圖1.4(a) 經由碎形化得到一較真實的海岸線圖(b)。

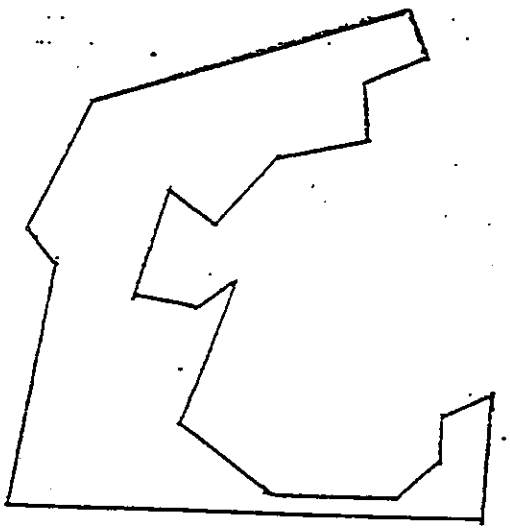
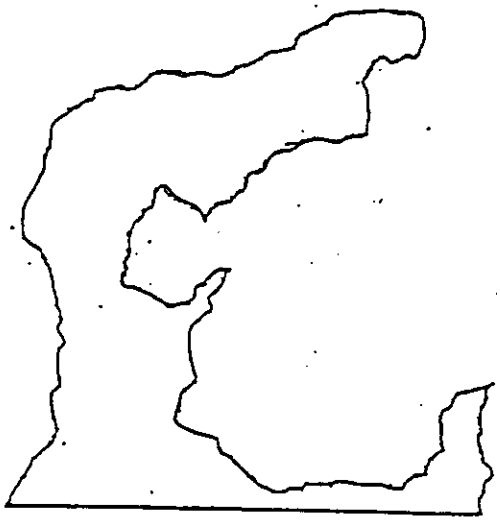


圖1.4(a) 粗略的海岸線圖



(b) 碎形化后的海岸線圖

(2)二維碎形法(Two-dimensional Fractal)

二維碎形法是將一維碎形法的處理對象一線段，擴充到二維的面(Polygon)，處理的理念上是相同的，都是分割粗略的處理元件(Primitive)成許多細小的元件，這也就是碎形化的過程。

目前表示三維物體的表面大都使用二維的面，二維元件(Two-dimensional Primitive)最常使用三角形平面或四邊形平面[5]，因這種平面(Polygon)的分割演算法較易於實行，其中三角形平面最簡單，同時不必考慮共面及平面的凹凸(Concave or Convex)問題。如圖1.5依1.5(a)之分割方式逐步細分便得到(b) 的各種不同亂度及細分程度

之零碎曲面(Fractal Surface)

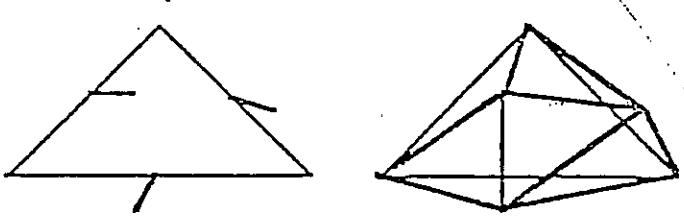
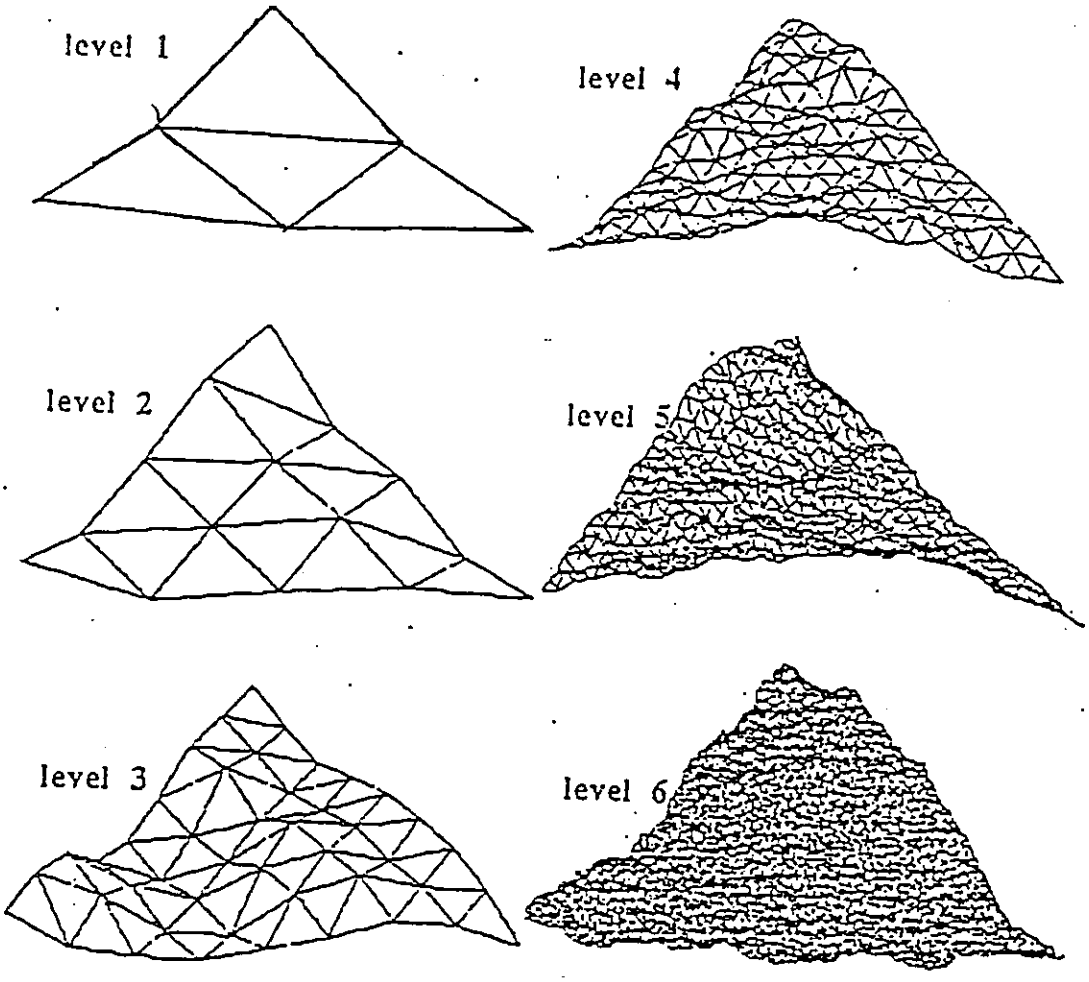


圖1.5(a)三角形平面之分割方式



(b)各種層次的細分情況

(3)三維碎形法(Three-dimensional Fractal)

就微觀角度言之，自然界中的物體均可由許多細小的元件組構而成，這些元件可小至分子、原子之類，基本上均可視為三維的微小物體，而三維碎形法的處理對象就是將三維的自然物體依體積切成許多微小的元件。但是欲達到世界中物體的相似，恐怕目前很難由電腦實行。另外IBM華特生研究中心(Watson Research Center)的愛倫、諾頓(Alan Norton)以疊代函數(Iterative Function)的遞迴方法(Recursive Method)實施於複數空間[6]，並用陣列處理器(Array Processor)進行實驗模擬得到很多奇形怪狀的四度空間物體，誠然不簡單，但這些物體，不但不易了解，而且，真實的世界中並不存在，故僅供研究。

(4)四維碎形法(Four-dimensional Fractal)

由於碎形法只能表現靜態物體的狀況，欲達到動態的運動方法，尚須加入時間函數(Time Function)，亦即引入模型運動(Model Motion)的觀念，更能有效地掌握複雜而且不規則的物體移動，像飄落的葉子，其軌跡絕非一條直線，它必會受到風力、重力及葉子本身形狀的影響，使其軌跡左右擺旋。

第四節 數學上效果的控制

前一節所述之方法乃是針對處理對象之不同而分類，至於物體表面的零亂效果(零碎且不規則)，需由布朗運動(Brownian Motion)來達成。對於自然物體的碎形特性，愛因斯坦(Albert Einstein)以數學力學的方法分析自然界中不規則運動的隨機現象，建立一套布朗運動的理論模型。可用以解釋自然物體運動的行為特性。由愛因斯坦的理論基礎孟德爾博(Mendelbrot B.B.)及范尼斯(Van Ness)[1]導出零碎的布朗運動 (fractional Brownion motion 簡稱fBm，爾後傅尼爾等人(Forniner et al) 以簡單而淺顯的中點替換法(Mid point Displacement Method)[4]來近似模擬複雜且隨機的布朗運動，更何況中點替換法的遞迴性，非常適合電腦執行。分析整個中點替換法有三個重要的調整變數：

<1>零碎次元(Fractal dimension)

控制碎形物體表面零亂的程度，零碎次元(D)愈大表示物體表面愈粗糙(Rough)，於是形狀變化愈大，表面積愈大，物體的幾何維度便愈高，反之物體的外形愈平滑，則幾何維度愈小，甚而有退化之情況發生，如坎特集合(Cantor dust)。圖1.6(a)詳細顯示不同零碎次元所產生

不同的粗糙程度。圖1.6(b)則是零碎次元退化的一種情況

。

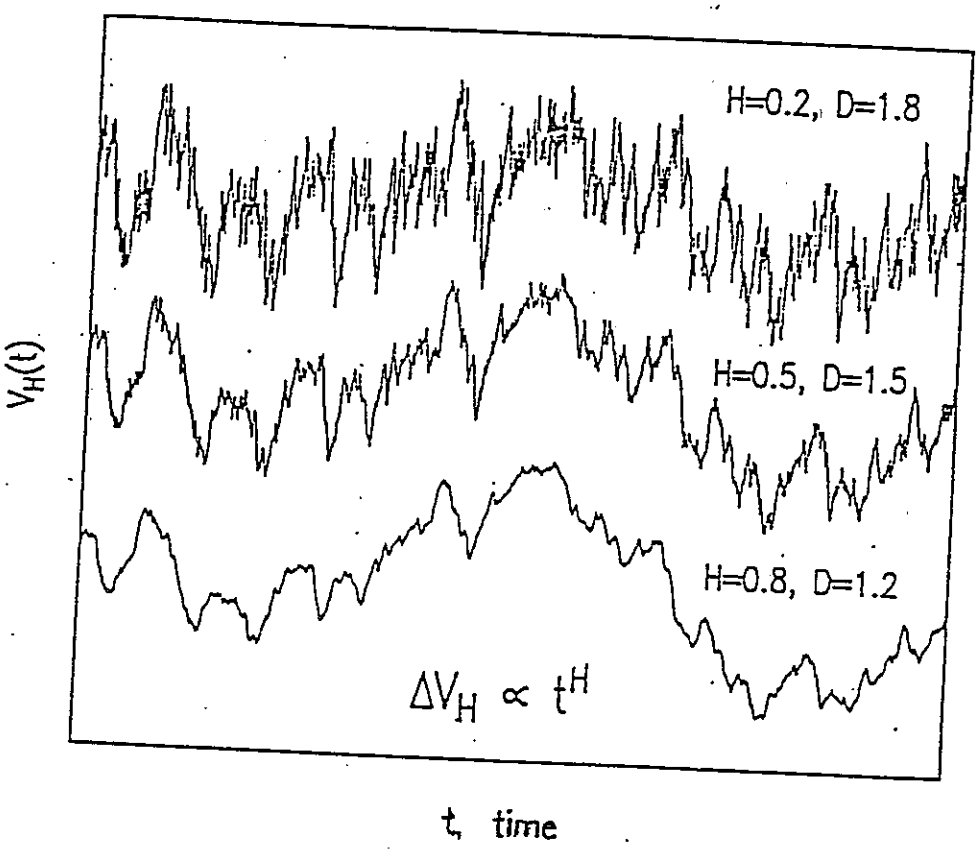


圖1.6(a)不同零碎次元產生不同之粗糙面

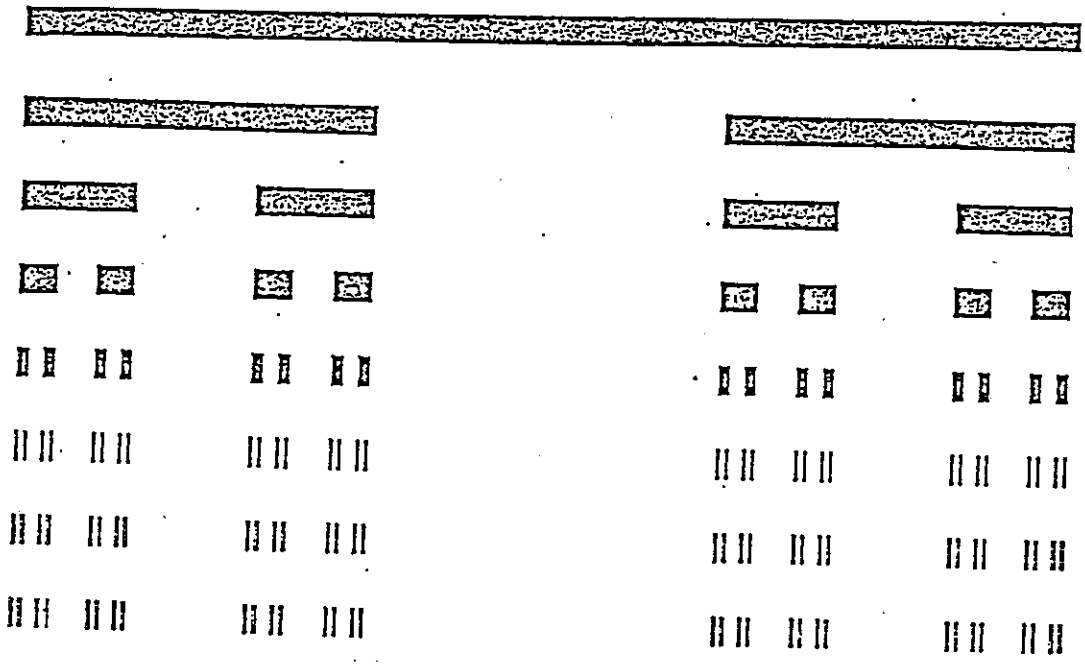


圖1.6(b)坎特集合一零碎次元的退化情況

<2>分配函數(Distributive Function)：

為了模擬零亂的效果，勢必由亂數產生器(Random Number Generator)取得亂數樣本，但又怕亂數可能會破壞物體原來的外形，以致造成不合理的物體，完全喪失真實感。所以必須藉由分配函數來控制亂數的產生，使其具收斂性(Convergence)。

<3>細分程度(Finer level)

人的眼睛受限於視力及眼界，無法明辨各種尺度的物體，例如在一公里處看山，也許可以看清楚山的凹凸起伏

，甚至一草一木；但若在100公里外看山，最多也只能看出山之輪廓。所以針對不同的尺度施以合適的細分程度，能降低不必要的計算，又可滿足使用者的需求。

以上三種重要的觀念會在第二章做更詳細的解釋；以使一般使用者均能了解各種調整參數的物理意義及調整方法，輕易地調整變數值，使物體外形作不同程度的變化，來符合不同使用者之需求，達到真實化的模擬效果。同時對各種自然碎形物體作一小部分的分類工作，建成以物體導向(Object Orient)為架構的使用者介面(User Interface)，不但讓後續研究者輕易接手，並為完整作畫系統，提供這部分的功能及可用性。

第五節　研究範圍

本論文研究之重點，希望能以高斯分配函數(Gaussian Distributive Function)為基礎[7]，控制亂數的產生，同時利用中點替換法來模擬布朗運動，讓使用者經由使用者介面(Interface) 調整零碎變數的值，得到所要的畫面。在本論文中嘗試使用不同的中點替換法來滿足自然物體的零碎效果。

接著在第二章中，將對布朗運動及分配函數的控制方

法，作較深入的探討、同時引出零碎幾何學的理論基礎及零碎特性的物理意義，然後再解釋中點替換法，如何能符合零碎的性質。第三章將討論整個作畫系統的架構及處理流程，並針對動畫者(Animator)系統的需求。依物体的維度，施於適當的碎形化處理，亦即將使用者輸入的概略物体外形作碎形化，得到零碎細緻的碎形物体。第四章是本研究的一些實驗結果，最後對本研究作一結論，並提及未來碎形物體的研究趨勢。

第二章　理論背景

碎形幾何學的基本現象早在十八世紀便被植物學家提出，植物學家們發現樹葉的成長過程就符合碎形特徵，他們觀察葉細胞成長是一種零亂的擴張運動，細胞不斷地往外延伸，但卻收斂在本身成長基因所控制的形狀，不會因零亂的生長而使一株樹長出外形差異太大的葉子。這種零亂的現象，直到1828年才由植物學家羅伯、布朗(Robert Brown)推廣到各類自然科學，他發現這種零亂的運動不僅僅存在於植物界，幾乎涵蓋所有自然科學研究的自然現象。往後的研究者不斷地確定布朗運動的適用性，如波林(Perrin J.) [8, 9]以布朗運動來解釋化學上的分子運動以及湯普森 (Thompson, D'A.W.) [10]將布朗運動應用在各種自然生物的成長型態(Growing Form)。研究布朗運動在1905年到1909年間達到巔峰，在理論建立方面，由愛因斯坦集其大成，在實驗應用方面，波林的貢獻最多。本章先對碎形幾何的自然特性作一介紹，再以布朗運動解釋各種自然特性的形成因素，並以正規分配函數(Normal Distribution Function)作為取樣之依據。

第一節　碎形幾何學(Fractal Geometry)

孟德爾博(Mandelbrot B.B.[3])研究複雜自然物體的結構並引出一門新的幾何學—碎形幾何學，能有效地描述複雜自然物體之結構。他展示了在碎形幾何下所製造的物體，他稱這些物體為碎形物體(Fractal Object)，但何謂碎形？它和傳統上常用的歐氏幾何表示法有何差異？下列幾點歸納了碎形表示法和歐氏表示法的主要不同點。

(1)歐氏幾何是傳統的用法，早在二千年前便已發展，而碎形幾何的發展卻是近十幾年的事。

(2)歐氏幾何有有一定特徵放大率(characteristic size or scale)。而碎形幾何則沒有所謂的特徵放大率，它具有自我相似性(Self-similar)及獨立尺度(Independent of scale or scaling)。

(3)歐氏表示法能精確地描述人造物(Man-made Objects)但卻不適合描述自然物的形狀，同時會產生麻煩且不正確的描述。在我們傳統形狀語言(Language of Shapes)的限制下，在機器生產物和自然物之間，多多少少會有一些固定的明顯差異。故歐氏幾何能輕易地描述及建構整齊而規則的人造物，而碎形幾何則能提供自然物體的表示法，讓電腦圖形中注入自然的特色。

(4)歐氏幾何均用數學公式(多項式)來描述物體外形，但碎形幾何用電腦語言中的程序(Procedure)或演算法(Algorithm)來建構，最重要的精神在於這些碎形幾何的程序或演算法均含遞迴性(Recursion)。

在傳統的歐氏幾何中，點為零維，線為一維，面為二維及物體為三維。但在碎形幾何中，線是具有無限振動(Wiggles)的變化，稱為碎形曲線(Fractal Curve)，由於無限振動將會幾乎佔滿某一塊面積而趨近於二維此乃無限振動的變化使曲線長度增加到無限大，故此碎形曲線無法由一維完全描述，但又未佔滿完整的面積，故維度小於二維，稱此分數維度為零碎維度。

關於碎形物尚有一重要特性—自我相似性，即物體的小部分的結構和整個物體的結構相似。最常被用來解釋零碎性的是著名的卡卻曲線(Koch Curve)，卡卻曲線的產生程序即為基本的碎形維度圖像的產生方法。圖2.1便是其產生的過程。

2.1.1 自然相似性及碎形維度(Self-Similarity and Fractal Dimension)

自我相似性及碎形維度其實是一體的兩面，碎形維度

是為了量度自我相似的形狀所產生的，而自我相似性卻是藉由碎形維度做為零亂的標準。一般而言，有限集合的自我相似性定義為：

定義：有限集合S為N個互相不重複的部分所組成的集合，當各部分集合與整個S同時進行相似轉換得到結果一致時，S即具有自我相似性。

像這種自我相似的圖形並沒有所謂的特徵放大率，亦即無論怎樣放大或縮小，所見到的圖形性質都一樣。在這種自我相似的觀念中，引入無限(Infinite)之概念，使得破碎維度容易了解，今以卡卻曲線為例說明以上所提之碎形理論的中心思想。

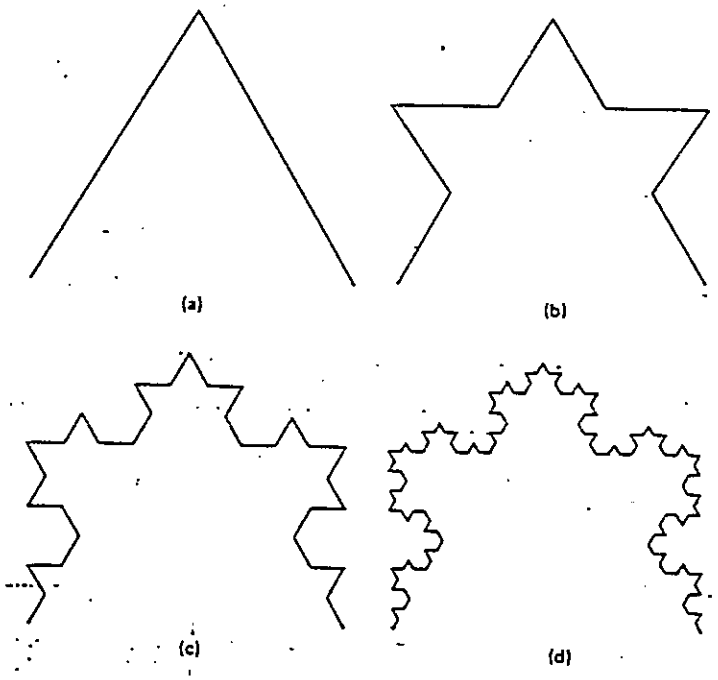


圖2.1 卡卻曲線的產生過程

如圖2.1所示以規則圖案產生碎形曲線，自圖中(a)含有兩條線段的曲線開始。首先將原來曲線圖樣縮小三分之一，並用以取代原圖中每條線段的中央三分之一部分得到(b)，再把縮小的(a)再縮小1/3，取代(b)中各線段中央的1/3的線段得到(c)，反覆將原圖縮小1/3，並用縮小圖修改上一次產生曲線中的線段，在各步驟中，曲線的線段數目以4的倍數增加，且曲線中各線段長度為原來長度的1/3，故每步驟的總長度以4/3的倍數增加。見圖2.2。

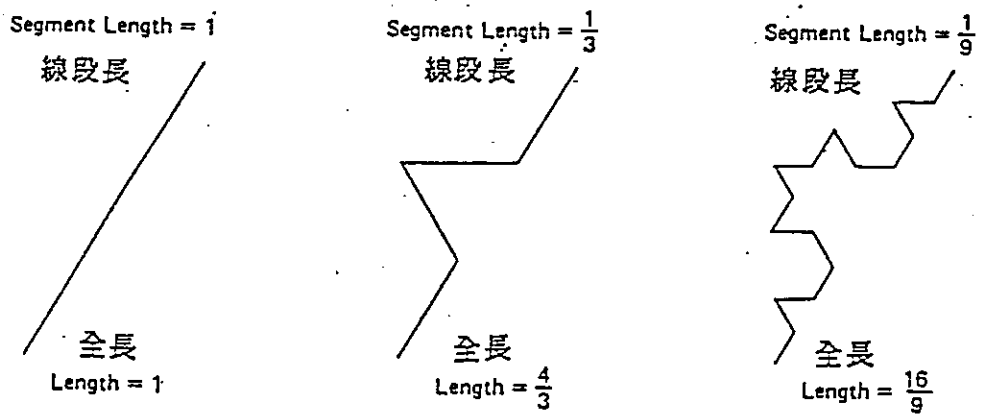


圖2.2 總長度每次增加原先的4/3倍

因此當碎形曲線愈分愈細時，曲線長度趨近無限大。可知原本長度有限的曲線經碎形化後，竟無法量度其長度，故無法用一維量度此碎形曲線，須用一維碎形維度來表示，由下列公式計算可得：

$$D = \log N / \log(1/S) \quad \ldots\ldots (2.1)$$

其中N是每一步驟中所細分的段數，每段都具有自我相似性，S是縮小倍數的指標。就上面的例子而言，線段被分割為4段，N＝4，而且使用的縮小倍數指標S＝1/3，由公式(2.1)可得：

$$D = \log 4 / \log 3 = 1.2619$$

對於公式2.1的推演是來自歐氏空間，在一維空間中，作碎形分割時，若S＝1/N，則必保持其原來的長度及維度(因D＝logN/log(1/S)＝logN/logN＝1，故維度不變，且增加的段數的係數剛好和縮小倍數相乘為1，故長度不變)。同理將二維曲線分成N等分而不改變其面積，$N = 1/S^2$，則必維度不變。而三維物體亦如此，可得$N = 1/S^3$，由此規則可推出一個普遍通用的公式(2.2)

$$N = \frac{1}{s^D} \quad (2.2)$$

將公式2.2作一轉換，得到求碎形維度的通用公式(2.3)

$$D = \log(N)/\log(1/S) \quad \ldots\ldots (2.3)$$

2.1.2 統計式自我相似性(Statistical Self-similarity)

上一小節所得到的卡卻曲線，在自我相似性的前提下產生規則的圖案，雖有細碎的效果，卻無零亂的形狀，這和零碎的自然景觀如海岸線尚有一段差距，為了加入零亂的因素，需用一亂數產生器來擾亂每次變化的大小，例如，

一海岸線長度為

$$\text{Length} = S \cdot N(s)$$

此N(s)為某種分佈下取樣{N: $N = 1/(S^D)$}的平均值，故

$$\text{length} \approx s \cdot \frac{1}{s^D} = \frac{1}{s^{D-1}}$$

則得一較真實的海岸線，其維度大約在1.15和1.25之間，相當接近1.2619，加入這種統計式觀念有下列好處：

(1)得到一經過擾亂的零碎形較符合自然現象。

(2)可以用統計觀念中的亂數分佈函數來控制物體外形之變化。

為了要達到統計式自我相似性及碎形維度等等碎形特性的需求，下節將引入符合自然規律的布朗運動。

第二節　布朗運動(Brownian Motion)

愛因斯坦所建立的布朗運動的理論，以數學力學的方

法分析這種隨機且不規則變化的運動。這種布朗運動完全表現零碎形幾何學的所有特性。布朗運動過程的定義如下：

定義1:所謂布朗運動即為具有如下性質的機率過程

$\{X(t)\ ;\ t \geqq 0\}$

(a)所有的增量$X(t+s)-X(s)$呈平均值為0($X=0$)，分散值(variance)為ct($\mu = ct$)的正規分配(Normal Distribution)

且$C>0$為固定的常數參數。

(b)在$n=2$的情況下，對於未重合的任意個時間間隔$[t_1\ ,\ t_2\ ]$及$[t_3\ ,\ t_4\ ]$(若$t_1 < t_2 < t_3 < t_4$)，增量$X(t_4)-X(t_3)$與增量$X(t_2)-X(t_1)$為具有(a)所示的分配獨立機率變數。當n為任意正整數時，對於不相交的n個時間間隔也是有一樣的現象。

定義2:正規分佈(Normal Distribution)

$$P(x,\mu) = \frac{1}{\sqrt{2\pi\mu}} \exp\{-\frac{(x-\bar{x})^2}{2\mu}\} \qquad (2.4)$$

公式2.4中μ為分散值，且定義(a)中的常數$C=1$。

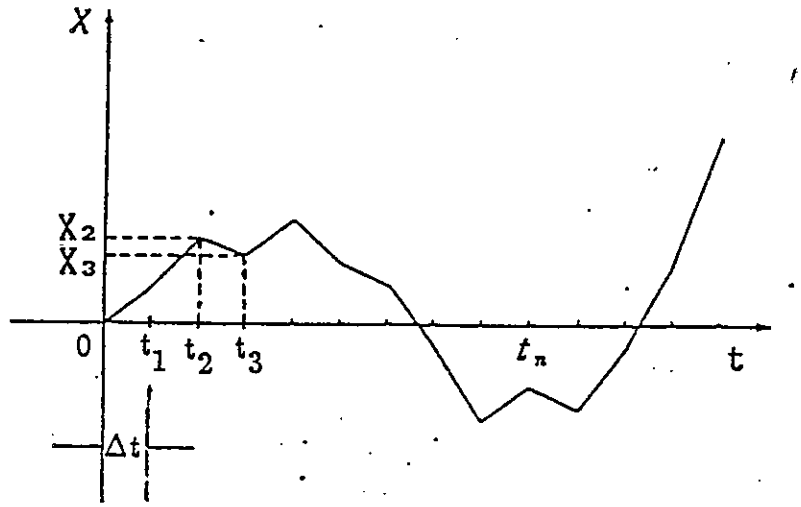


圖2.3 布朗運動的軌跡

依循上述定義1.2我們的推導得到布朗運動的軌跡，以下便是推導過程：

<1>從初始狀態t=0開始求運動軌跡。$t \in [0, t_3]$，若定義1(a)假設 S=0，$t=t_3$，可知$B(t_3)-B(0)$呈平均為0，分散值為t_3（因c=1)的正規分佈。如圖2.3所示，若b(0)=0，$t=t_3$ 時的機率密度函數P_3(propability Density Function)可從定義2得到，如下所示：

$$P_3 = \frac{1}{\sqrt{2\pi\, t_3}} \exp\left(-\frac{x^2}{2t_3}\right)$$

$$= \frac{1}{\sqrt{6\pi\Delta t}} \exp\left(-\frac{x^2}{6\Delta t}\right) \qquad (2.5)$$

(因 $t_3 = 3\Delta t$)當 $t=t_3$ 時的變位B可機率性地求得。

<2>從 $t=t_2$ 的狀態開始求運動軌跡 $t \in [t_2, t_3]$，

$t=t_2$ 的機率密度函數 P_2 以和(1)完全相同的方式處理，則得到：

$$P_2 = \frac{1}{\sqrt{4\pi\Delta t}} \exp\left(-\frac{x^2}{4\Delta t}\right) \qquad (2.6)$$

(因 $t_2 = 2\Delta t$)故 $t=t_2$ 時的變位 X_2 可機率性地求出。

其次想要求出的是 $t=t_3$ 時變位 X_3，但這時應注意的是 $t=t_3$ 時的機率密度函數 P_3，在這個時候由定義(a)可知呈現一個以 $x = X_2$ 為中心，分散值 $=\Delta t$ 的正規分佈。也就是把定義(a)中的S定為 t_2。

因 $t=t_2$ 時的變位 X_2 是從以X=0為中心的正規機率密度函數求出，所以想要求出 $t=t_3$ 的機率密度函數值做對應於 $X_2 = -\infty \sim \infty$，之間的各 X_2 值的機率密度函數做為加權值，乘上正規分佈函數（平均值 $X=X_2$，分散值 $=\Delta t$)之後再對 X_2 作積分工作，如下所示之計算過程：

$$P_3 = \int_{-\infty}^{\infty} \underbrace{\frac{1}{\sqrt{4\pi\Delta t}} \exp\left(-\frac{x_2^2}{4\Delta t}\right)}_{\text{加權函數}} \cdot \underbrace{\frac{1}{\sqrt{2\pi\Delta t}} \exp\left\{-\frac{(x-x_2)^2}{2\Delta t}\right\}}_{\text{正規分佈函數}} dx_2 \qquad (2.7)$$

(平均=0，分散值 $t_2 = 2\Delta t$)得到

$$P_3 = \frac{1}{\sqrt{8\pi\Delta t}} \int_{-\infty}^{\infty} \exp\left(-\frac{s^2+2(x-s)^2}{4\Delta t}\right) ds \qquad (2.8)$$

$(X_2 = S)$

$$P_3 = \frac{1}{\sqrt{8\pi\Delta t}} \int_{-\infty}^{\infty} \exp\left(-\frac{s^2+2x^2-4xs+2s^2}{4\Delta t}\right) ds$$

$$= \frac{1}{\sqrt{8\pi\Delta t}} \exp\left(-\frac{x^2}{2\Delta t}\right) \int_{-\infty}^{\infty} \exp\left(-\frac{3s^2-4xs}{4\Delta t}\right) ds \qquad (2.9)$$

考慮積分符號中的指數部分則

$$-\frac{3s^2-4xs}{4\Delta t} = -\frac{3}{4\Delta t}\left(s^2-\frac{4}{3}xs\right)$$

$$= -\frac{3}{4\Delta t}\left\{\left(s-\frac{2}{3}x\right)^2-\frac{4}{9}x^2\right\}$$

若 $S \cdot 2X/3 = \mu$，由 $du=ds$ 可得

$$P_3 = \frac{1}{\sqrt{8\pi\Delta t}} \exp\left(-\frac{x^2}{2\Delta t}\right) \int_{-\infty}^{\infty} \exp\left\{\left(\frac{3}{4\Delta t}\right)\frac{4}{9}x^2\right\} \cdot \exp\left(-\frac{3}{4\Delta t}u^2\right) du$$

$$= \frac{1}{\sqrt{8\pi\Delta t}} \exp\left(-\frac{x^2}{2\Delta t}+\frac{x^2}{3\Delta t}\right) \int_{-\infty}^{\infty} \exp\left(-\frac{3}{4\Delta t}u^2\right) du \qquad (2.10)$$

此處若 $\sqrt{\frac{3}{4\Delta t}}\,u = y$ 則 $\frac{3}{4\Delta t}u^2 = y^2$

而 $du = \sqrt{\frac{4\Delta t}{3}}\,dy$ 及 $\int_{-\infty}^{\infty} \exp\{-p^2\}\,dp = \sqrt{\pi}$

因此

$$P_3 = \frac{1}{\sqrt{8\pi\Delta t}} \exp\left(-\frac{x^2}{6\Delta t}\right) \int_{-\infty}^{\infty} \exp\{-y^2\} \cdot \sqrt{\frac{4\Delta t}{3}}\, dy$$

$$= \frac{1}{\sqrt{8\pi\Delta t}} \sqrt{\frac{4\Delta t}{3}}\, dy \sqrt{\pi} \exp\left(-\frac{x^2}{6\Delta t}\right)$$

$$= \frac{1}{\sqrt{6\pi\Delta t}} \exp\left(-\frac{x^2}{6\Delta t}\right) \qquad (2.11)$$

公式(2.11)為平均=0，分散值$t_3 = 3\Delta t$的正規分佈函數，與公式(2.4)完全相同。因此$t=t_3$時的變位X_3可用(1)或(2)兩種方法求出。當然也可以從$t=t_1$開始的狀態求出。

同樣地，任意時間tn時的軌跡求法也可以知道了，即某些時間的機率密度函數，可在定義1(a)中的S設定為該時間之前的某時刻之後求出，此時運動軌跡只和時間 t=s 有關，而與過去的狀態無關，故布朗運動也可以說是馬可夫（Markoff)程序的一種。

整理上述步驟，可得到以下具體的結果：

(1) 決定定義1(a)中的C及時間間隔Δt，並以$C\Delta t$為分散值。

(2) 把一樣的亂數經由正規分佈函數得到平均值為0，分散值為C△t的正規分佈亂數。

(3) 以 t ＝0時X＝0，並以 t ＝△ t 時的變位X_1 當作(2)中的亂數求出。

(4) 把 t ＝2△ t 時的變位X_2 當作X_2 =[(2)中的亂數]+X_1 ，求出。

(5) 把 t ＝3△ t 時的變位X_3 當作X_3 =[(2)中的亂數]+X_2 ，求出。

(i+2)同理把 t ＝ i △ t 時的變位X 當做X_i＝[(2) 中的亂數]+X_{i+1}，求出。

這個方法最大的問題在於一正規布佈亂數的求得。我們將在下一節對此亂數控制的問題，深入探討。

其次要簡單地介紹孟德爾博及范尼斯[1] 共同導出的一碎形維度的布朗運動 (Fractional Brownian Motion，簡稱fBm)。在上述布朗運動軌跡的求法中，為了簡化起見，以公式(2.4) 做前提假設。一般而言，碎形維度的布朗運動除了正規分佈之外，還對某隨機函數之所有值的集合w擴充，B(t,w)表示這種布朗運動，且t表示時間是一實數參數，$-\infty < t < \infty$。

如果B(t,w)是一增量為獨立高斯分佈的亂數函數，其

增量平均值為0，[i.e.uB(t+△μ,w)- B(t,w)=0] 且分散值為$|t_2 - t_1|$,(t_1 ,t_2)與(t_3 ,t_4)並不相交時，若B(t_2 ,w)-B(t_1 ,w)與B(t_4 ,w)-B(t_3 ,w) 沒有依存在，這種運動稱為正規布朗運動(Ordinary Brownian Motion)。

定義3：當0<H<1,且b_0 為任意實數，在t=0時的初值，則對於t>0而言,布朗運動 $B_H(t,w)$ 定義如下：

$$B_H(0,w)=b_0$$

$$B_H(t,w) - B_H(0,w) = [\frac{1}{\Gamma(H+0.5)}]$$

$$\{\int_{-\infty}^{0}[(t-s)^{H-1/2}-(-s)^{H-1/2}]dB(s,w)+\int_{-\infty}^{0}(t-s)^{H-1/2}dB(s,w)\} \quad (2.12)$$

$B_H(t,w)$為使用偏差$(t-s)^{H-1/2}$ 時得到的$B_H(t,w)$的運動平均，我們稱此亂數函數$B_H(t,w)$為分碎布朗運動(Reduced fBm)。當H=0.5時公式(2.12)即成為正規布朗運動。在分碎布朗運動的定義中，有一參數非常重要，就是H,值域為(0,1),它代表很多意義，從物理的角度來看,H值代表布朗運動的亂度,也就是物體外形零碎變化的程度,見圖1.6(a)，當H值愈接近1則亂度愈小，物體表面愈平滑，零碎維度愈接近物體原維度(當H=1時,物體表面無變化，零碎維度等於物體的歐氏幾何維度)。反之，當H值接近0，則亂度愈大，物體表面愈零碎，零碎度愈增，(當H=0時，物體表面變化完全，零碎維度等於物體幾何維度加1)。

在機率統計的角度來看[11]，H值表示亂度函數$B_H(t,w)$ 產生增量的依存性，亦即亂度函數 $B_H(t,w)$ 產生的增量呈現一種分配函數，當H=0.5,$B_{0.5}(t,w)=B(t,u)$ 為獨立高斯

分佈(Independent Gaussion Distribution)平均值為0，分散值為△t，當H值趨近於1時，分佈函數的相關係數 C>0，則對分佈函數中的亂數X_i,X_{i+1}產生以下影響：

$$C=\frac{Cov[X_i,X_{i+1}]}{\sigma_{X_i}\cdot\sigma_{X_{i+1}}} \quad , \quad E[(X_i-\mu)(X_{i+1}-\mu)]>0$$

故可得$(X_i-\mu)$和$(X_{i+1}-\mu)$須同為正或同為負，

因此可判定 X_i，X_{i+1} 正負號相同，差數不大，故可知物體外形受亂數擾亂後，變化較低。反之，H值趨於0時，分佈函數C<0，X_i 及 X_{i+1} 正負號不同，亂數變化大，物體外形就會愈零亂。

2.2.1 內部與外部的一致性(Internal and External Consistency)

在提出模擬碎形布朗運動 (fBm)的演算法之前，需先討論這種演算法的需求。首先，這種模擬碎形布朗運動演算法需考慮其效率，因為高品質的圖片，通常需要10^5 甚至10^6 樣本點。另外有２種類似的性質亦須考慮——內部及外部一致性：

(1) 內部一致性就是物體在任何詳細尺度及合適座標空間中的任一位置，均可被重新產生。如 圖1.4

(a)產生(b)，只要空間不變及亂數初值不變，則無論置於任何位置或任何時間，產生的結果永遠和(b)一樣。

(2) 外部一致性，就是當物體的二個碎形元件相連且共用一個共同邊界（Common Boundary)，我們必須考慮在任何尺度（Scale)下，此共同邊界上的連續性，如下圖2.4 便是兩相連的平面在碎形化後產生裂縫（Gap)。

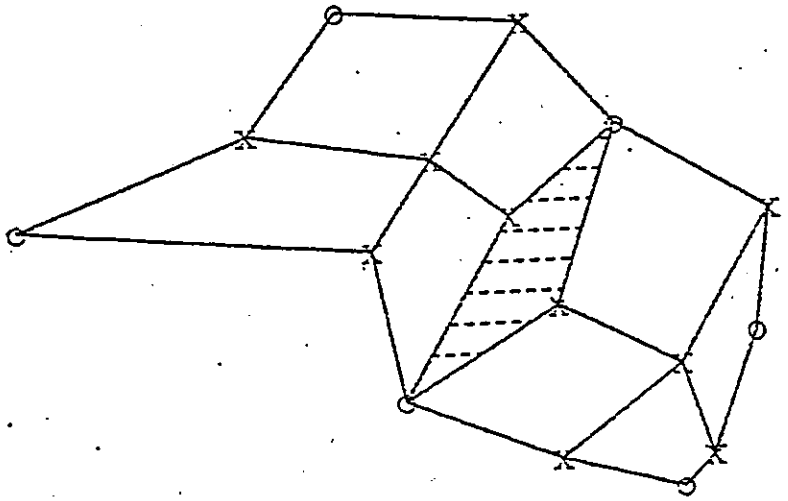


圖 2.4 兩平面間的共同邊界所產生的縫隙

2.2.2 孟德爾博計算 fBm 的方法

孟德爾博將計算 fBm 的方法分為三類：

(1) 剪替處理 (Shear Displacement Process)

在此法中孟德爾博用N維的分數波以松體(Fractional Poisson Field) 其中每一點P都相對一值F(P)，這個值F(P)是收集無限次步驟的總和，每一步驟分成三部分：方向 (Direction)、位置(Location)，振幅 (Amplitude)；每一部分都是互相獨立的亂數變數所產生的序列 (Sequence)。這方法常被用以模塑海岸線之類的碎形自然物。

(2) 修正的馬可夫處理 (Modified Markov Process)

此法須求兩個項目之和，一個是馬可夫一高斯處理 (Markov-Gauss Process)，屬於低層頻率 (Low-Level Frequency)；另一項為M次馬可夫一高斯處理的加權總和，屬於高層頻率(High-Level Frequency)。此法很逼近離散的零碎高斯雜訊 (Discrete Fractional Gaussion Noise),孟德爾博稱此法為快速零碎高斯雜訊(Fast Fractional Gaussion Noise)。

(3) 快速傅利葉轉換過濾法 (Fast Fourier Transfrom Filting)

此法是一個假亂數的產生器，能產生白雜訊 (White Noise)W(t)。而W(t)是經由快速傅利葉轉

換函數T(f)過濾而產生。此法常被利用以產生非常深刻（Impressive）生動的碎形圖形。

以上三種方法各有優劣，第一種的時間複雜度最高為$O(N^3)$，而第二個方法似乎只適用於一維碎形處理。而第三種方法有一主要缺失—就是資料樣本輸入困難。於是由孟德爾博的理論，傅尼爾、浮塞爾及卡本特等人(Fournier et al) [4]，提出一個遞迴分割法（Recursive Division Algorithm）稱為中點替換法（Mid-Point Displacement），能簡便地產生近似一維布朗運動的樣本路徑，產生圖形生動，且方法較有效率。此法有二個重點：

(1) 碎形布朗運動（fBm）是自我相似，且H是自我相似的係數。意即若以h^{-H}為因子，重新度量則 $B_H(t+\Delta t,w)-B_H(t,w)$及$B_H(t+h\Delta t,w)-B_H(t,w)$有相同的分佈（Distribution）。

(2) 1968年孟德爾博和范尼斯證明的事實，當$0\leq \mu \leq 1$，$B_H(0,w)=0$ 及 $B_H(1,w)=1$，$B_H(t/2,w)$ 的條件期望值為$1/2(t^{2H}+1-||\mu-1||^{2H})$。

第三節　亂數分佈的控制

前述亂數函數產生的增量為一高斯分佈，平均值為0，

分散值和時間差，或比例，表示如下：

$X(t_2)-X(t_1) \rightarrow$ 為高斯分佈

$E[|X(t_2)-X(t_1)|^2]$和 t_2-t_1 成比例。

首先，我們需要一個常式用以產生高斯亂數，此亂數需呈常態分佈N(0,1)，我們可以利用電腦提供的假亂數產生常式Rand()，通常此Rand()常式傳回的亂數Y為平均分佈(Uniform Distribution)的亂數，均勻地分佈在 [0,A] 的區間內，A值(A值為$2^{31}-1$或$2^{15}-1$)。

依照中央極限定理 (Central Limit Theorem) 取n個樣本Y1……Yn，均為相同分佈的隨機變數。

$$Z_n=\frac{\Sigma_{i=1}^{n}Y_i-E(\Sigma_{i=1}^{n}Y_i)}{\sqrt{Var(\Sigma_{i=1}^{n}Y_i)}}=\frac{\Sigma_{i=1}^{n}Y_i-\frac{n}{2}A}{\sqrt{\frac{1}{12}A^2}}=\frac{1}{A}\sqrt{12/n}\,\Sigma_{i=1}^{n}Y_i-\sqrt{3n} \quad (2.13)$$

當n>3，Zn 就趨近於常態分佈，由於我們需要的高斯分析為平均值等於0，分散值和時間差 ($|t_2-t_1|$) 成比例，故可將此時間差提出，或為一個可調整的變數—Sigma，它代表著亂數產生時分佈平均值附近的密集度。如下圖 2.5 是三種不同分散值的正規分布圖。

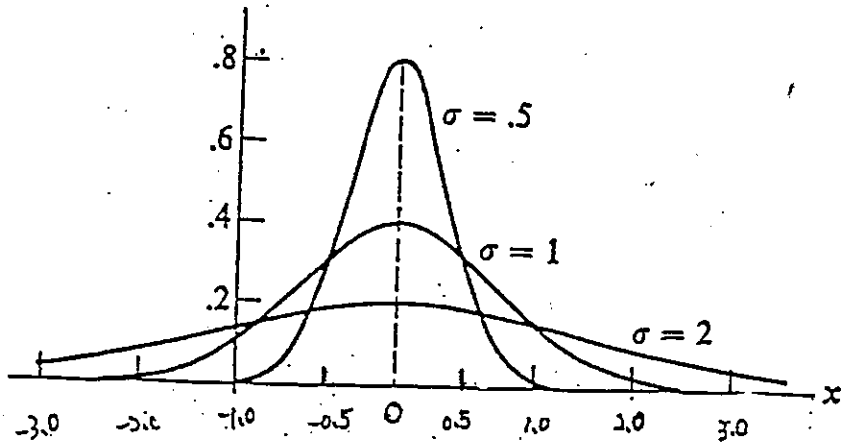


圖2.5 三種不同分散值的正規分佈圖

當 Sigma 愈大表示亂數分佈愈分散，當 Sigma 趨近於無限大時，為均勻分佈 (Uniform Distribution)。當 Sigma 愈小時，則相反，亂數向平均值等於零集中。這種集中的現象，表示亂數變化愈小，亦即隨機中點的位移量便愈小，對碎形物體的外形變化量而言有收斂的(Covergent)意味。

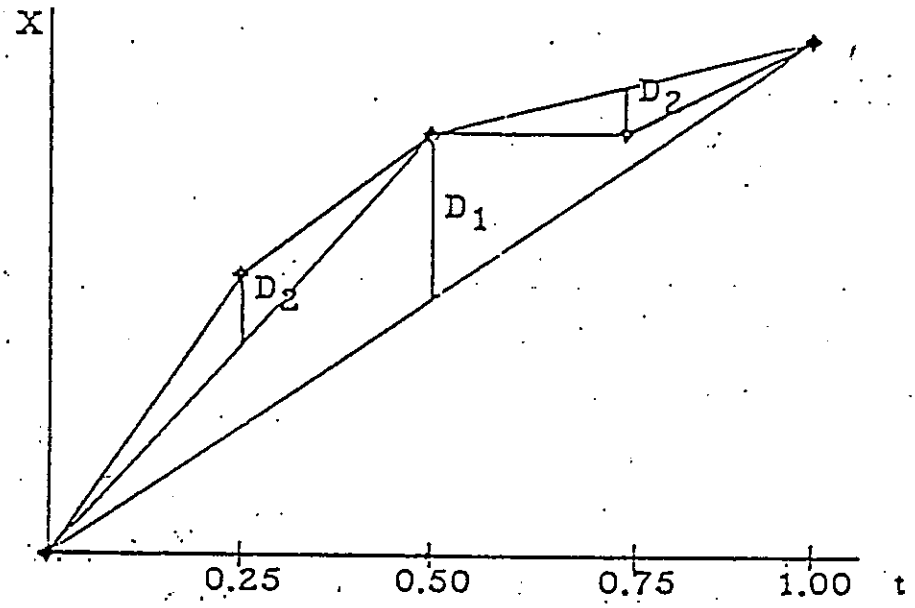


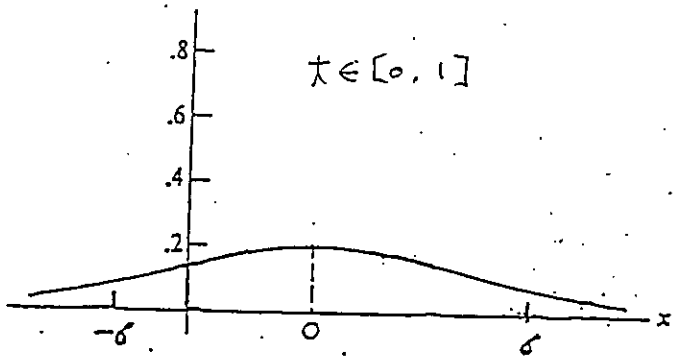
圖2.6 中點替換法的前二個步驟

依上節述，當 H=0.5時，零碎市朗運動為平均值等於0，分散值等σ^2 的高斯分佈。如圖2.6 假設時t [0, 1]，且X(0)=0，令 X(1) 為高斯隨機變數的樣本，此高斯分佈平均值為0，分散為$d^2=\sigma^2$，見圖2.7 level 0。依此分佈可得

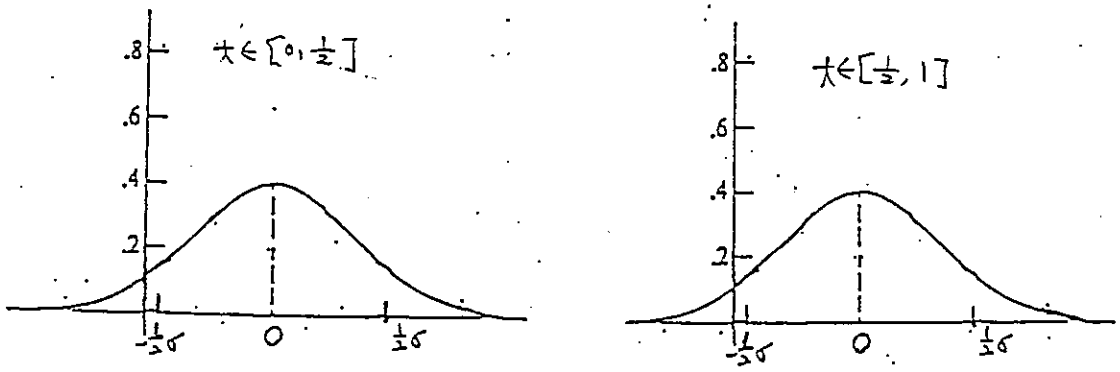
$$Var(X(1)-X(0))=\sigma^2$$

並期望能得

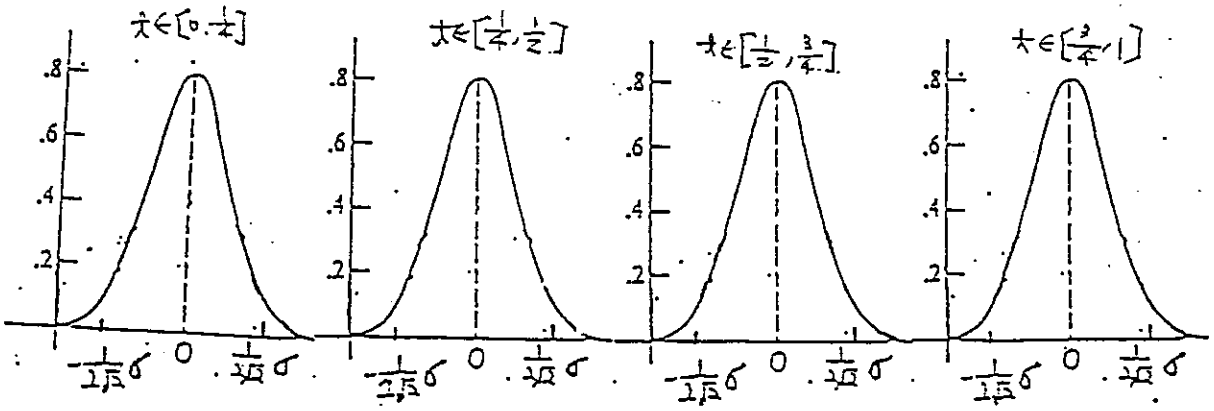
$$Var(X(t_2)-X(t_1))=|t_2-t_1|\sigma^2 \qquad (2.14)$$



level 0 $\quad d = \sigma$



level 1 : $d_1 = \frac{1}{2}\sigma$



level 2 : $d_2 = \frac{1}{2\sqrt{2}}\sigma$

圖 2.7 分割過程中，各隨機中點的分佈函數圖

此時$0 \leq t_1 \leq t_2 \leq 1$，令X(1/2)為X(0)和X(1)的平均再加一隨機位移量(offset)D_1 見圖2.6，並符合平均值為0，分散值為d^2 ：

$$X(1/2)-X(0) = (1/2)\ [X(1)-X(0)]+D_1$$

在自相似的前提下，X(1/2)-X(0)及X(1)-X(1/2)分別在[0,1/2]及[1/2,1]的區間中，具有相同的分佈函數，見圖2.7 Level 1，其平均值為0 ，分散值可由公式(2.14)求得

$$Var[X(1/2)-X(0)]=(1/4)\ Var[X(1)-X(0)]+d_1 =(1/2)\sigma^2$$

$$d_1 =(1/4)\sigma^2$$

重複上一步驟於 $t\in[0,1/2]$中得到 D_2 ，同理 X(1/2)-X(1/4)及X(1/4)-X(0)都是平均值為0，分散值為 d^2 的高斯分佈，且

$$Var[X(1/4)-X(0)]=(1/4)\ Var[X(1/2)-X(0)]+d_2 =(1/4)\sigma^2$$

$$d_2 =(1/8)\sigma^2$$

相同的理念可求出X(3/4)-X(1/2)及X(1)-X(3/4)的分佈函數。延伸上述步驟可得一規律性的公式

$$d_i^2=\frac{1}{2^{i+1}}\sigma^2 \qquad i=1,2,3... \qquad (2.15)$$

而相對應的時間區間△t為 2^{-n}。

依上述H=0.5推廣到 $H \in [0,1]$ 可得到對應於公式(2.14)的公式：

$$Var(X(t_2)-X(t_1))=|t_2-t_1|^{2H}\sigma^2 \tag{2.16}$$

$$d_n^2=\frac{\sigma^2}{(2^n)^{2H}}(1-2^{2H-2}) \tag{2.17}$$

我們可以輕易地了解 Var[(X(1/2)-X(0))]=Var[(X(1)-X(1/2))]=$(1/2)^2\ \sigma^2$，但公式2.16當 $t_2=3/4$, $t_1=1/4$ 時，Var[(X(3/4)-X(1/4))]不等於 $(1/2)^2\ \sigma^2$，形成不穩定增量，這是中點替換法的唯一缺失。

其次在分割過程中，各隨機中點的分佈函數提供一個重要的訊息——內部一致性(Internal Consistency)，依分割的Level愈大，亂數的分佈愈集中於0的附近，顯示位移量亦依Level的增加而變小，使物體外形具有收斂性(Convergence)，不致變形。

第四節　碎形物體分類架構

綜合前幾節討論，可將碎形自然物體依處理元件分類：一維碎形處理，二維碎形處理，三維碎形處理及四維碎

形處理如圖2.8所示：

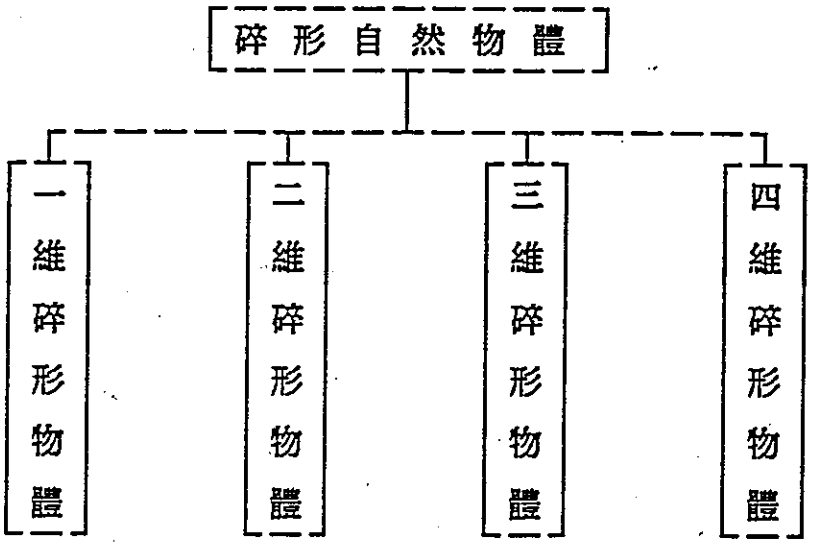


圖2.8 不同元件碎形自然物的分類

在人類可見範圍的自然物體包羅萬象，但除了一些像雲、雨、霧等三維以上，特殊且複雜度高的物體之外，大部分都可用一維或二維碎形處理來產生，故本研究限於設備及時間上的考慮，只對一維及二維碎形處理，作一些自然碎形物體適用性的研究，並提供使用者，調整物體外形零碎程度及變化的訊息。如下圖2.9所示分類方法：

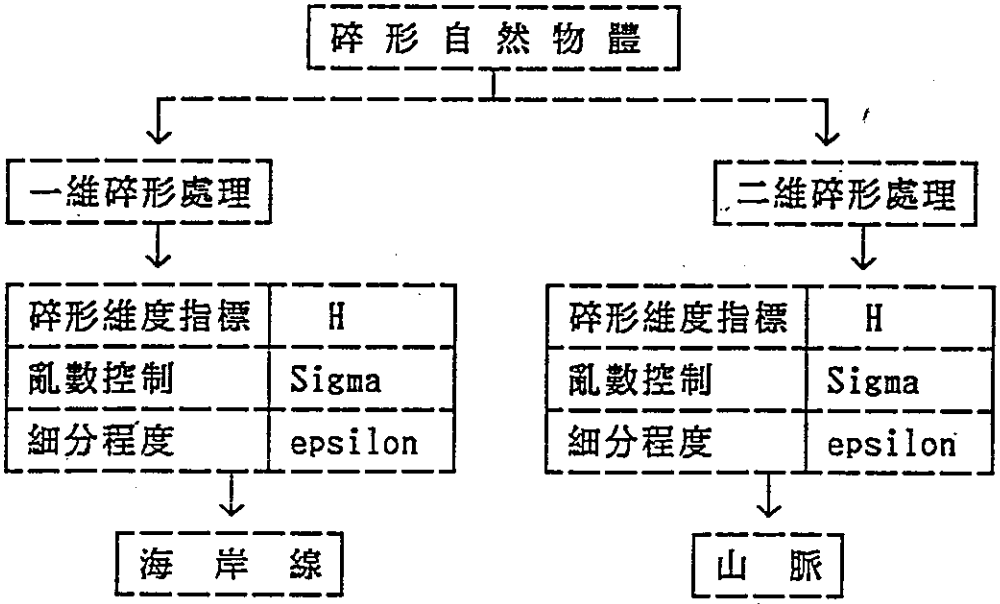


圖2.9 部分碎形自然物體分類架構圖

第五節 動態成長的模擬

本論文為了使碎形自然物能真實且生動地呈現在螢幕中，引入中繼圖(Inbetween) 的觀念，達到碎形然物動態成長的模擬效果。使用者需輸入第一張及最後一張畫面中物體的粗略形狀，並將第一張和最後一張的點互相對應，而互相對應的點，經由不同的直線或曲線路徑，產生各中繼圖的粗略形狀，並在產生中繼圖的過程中，馬上予以碎形化，供使用者檢驗是否合於其模擬效果，若不滿意可再調整各對應點的路徑。由於路徑的控制並非碎形理論的領域，並且同一研究群(researching Group) 的同學，已有許多位做主構線法(Key Frame Method)的研究，在此便不詳論。

第三章　系統架構與碎形化處理

本文在第一章曾提及增加動畫者系統功能，希望和自然物的碎形化處理方法連結成為一完整的碎形處理系統。首先本章將對動畫者系統和碎形化處理的連結作一介紹，其次說明碎形化處理的方法，並對自然物體的碎形化效果作一比較。

第一節　碎形處理系統架構

一般繪圖系統產生影像的流程大約分成三個部分：

(1)模塑物体外形

(2)物體前後排序(object sorting)及可見面處理(visible polygon process)

(3)物體柔化(rendering)

然而動畫者系統只對後面兩部分做適當處理，因此使用者須先塑模物體的外形後才能起動動畫者系統，如圖3.1所示。

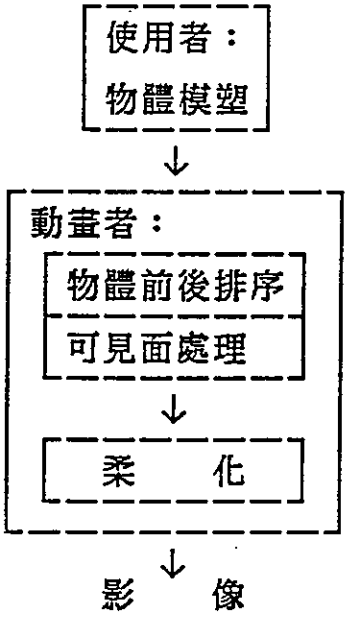


圖3.1 動畫者系統架構

前文曾提及，模塑人造物體大部分均可由多項式來表現，對使用者而言並不會太困難，但是欲模塑自然碎形物體確非使用者能力所及，因此本論文的目的希望以碎形化處理作為使用者和動畫者系統的碎形處理介面，如下圖3.2。

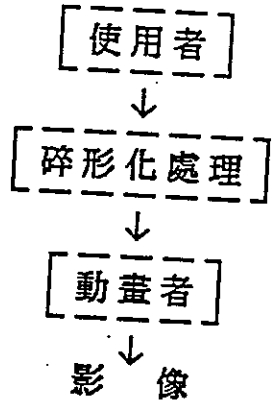


圖3.2 連結碎形化處理及動畫者系統

如此不但增加動畫者系統的模塑功能，同時亦方便使用者使用動畫者系統，使用者只要輸入粗略的物體形狀，經由碎形化處理可得一精細的物體形狀，並可調整碎形參數(fractal parameters)產生一個真實且合乎使用者需求的物體形狀。

由於動畫者系統僅接受固定格式的平面組合檔案(polygon file)，而且組合元件須為二維的凸多邊形平面(convex polygon)，因此碎形化所得的物體空間描述檔案必須符合動畫者所要求的平面組合檔案格式。如下圖3.3所示。

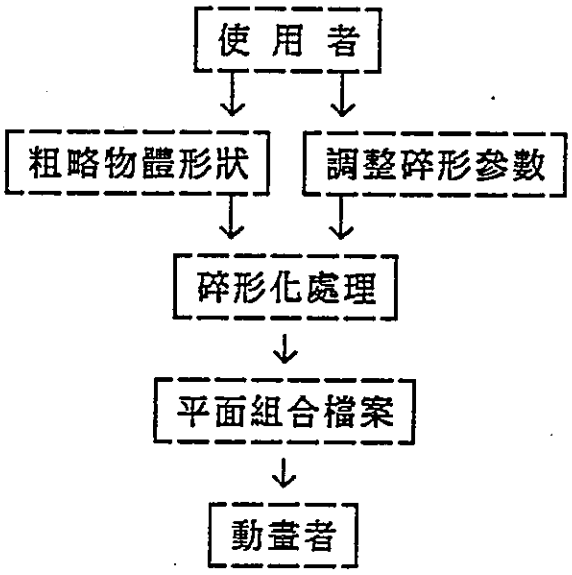


圖3.3 模塑物体碎形化的過程

另外本系統考慮到，動畫者系統產生影像相當緩慢，而且欲顯像至全像顯示器(raster displayer)須透過專用輸出軟体 —psout。吾人希望能有一軟体讓使用者迅速地看到碎形化後物体的形狀，剛好有threed這套軟体滿足需求，由於threed傳送資料的對象為向量式顯示器 (vector displayer)，故使用者可快速地看到由線條描繪的物體零碎外形。若使用者確定物体外形滿足自己的需要，便可將此物體輸入動畫者系統處理，產生真實的碎形自然物體。下圖3.4便是完整的碎形處理系統。

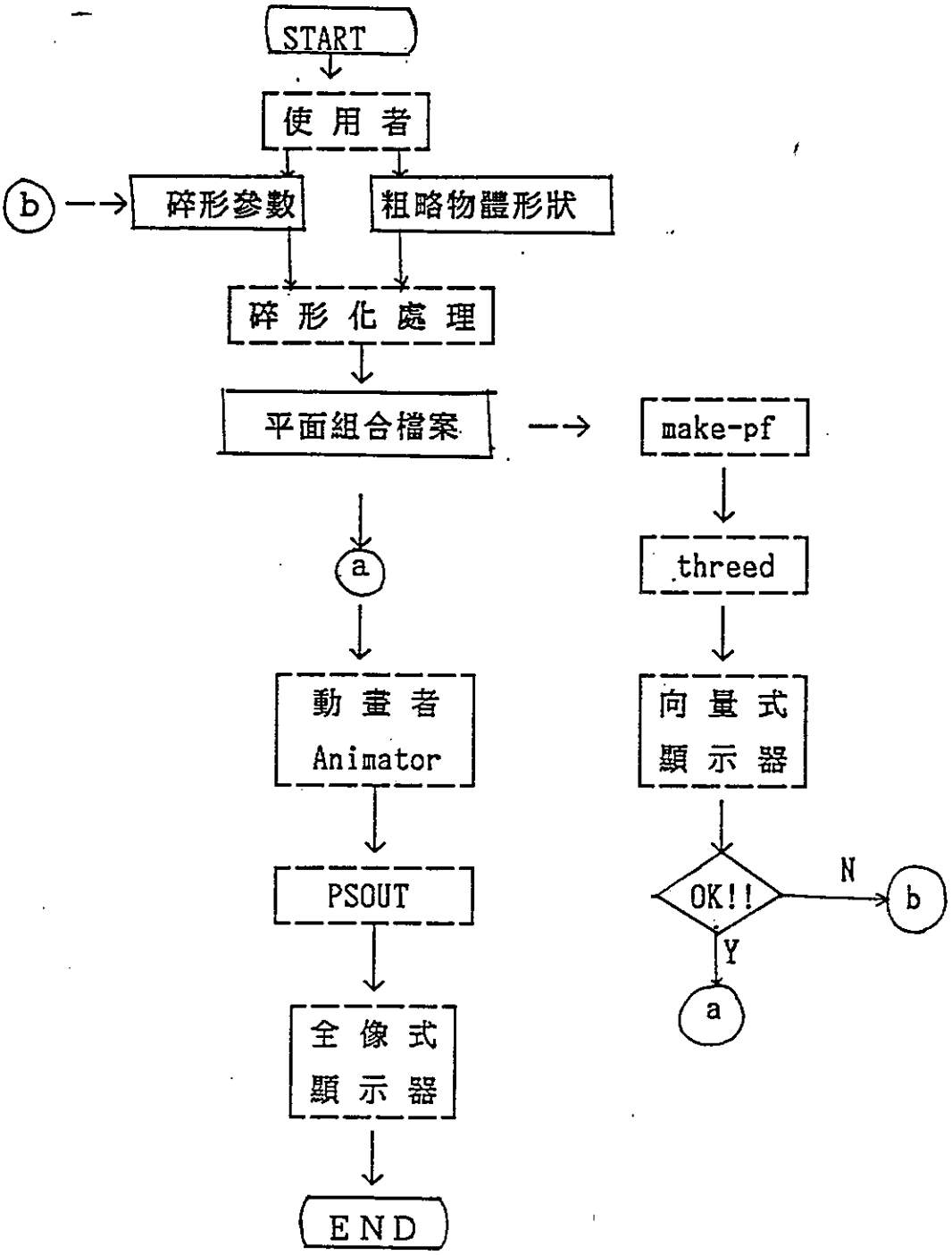


圖3.4 完整的碎形處理系統流程

第二節　碎形化處理

自然界的零碎特性可由布朗運動來模擬，一般而言最適於解釋布朗運動的模型是分子運動模型，如下圖3.5 為平面上分子運動的軌跡。

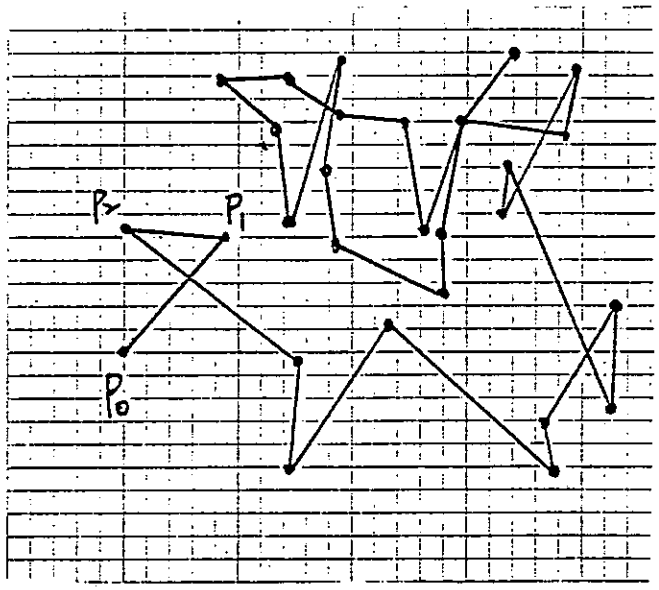


圖3.5 分子模型的布朗運動路徑

由圖3.5 中分子運動的情形可知，造成這種零亂的現象有二個重大因素：位移量(offset)及運動方向 (motion direction)例如圖3.5中，分子在P_0 位置移動到P_1 位置，方向為P_0 至P_1 ，移動量為$||P_1 - P_0||$ (距離)。有關位移量的隨機控制，已在前章第二、三節討論過 (由碎形維

度指標及高斯分佈函數控制）。故本節的重點在於方向隨機性的控制與其適用性的探討，並用中點替換法分別對不同維度的自然物體元件做碎形化處理，中點替換法以遞迴分割(recursive subdivision) 的方式分割物體元件達到細碎的目的，再以隨機分佈函數所產生的亂數擾亂細分後的小元件，得到零亂的效果。同時產生既細碎又零亂的自然物體。如圖2.6及圖1.5，分別對一維線段及二維平面作分割及零碎化的處理。

3.2.1 一維碎形處理－線段(polyline)

由於一維碎形法處理對象為平面上的線段，故擾亂的方向以線段的垂直法向量(normal vector of polyline)為中心向量(central vector)，有下列幾種方法：

(1)固定法向量(fixed normal vector)：

線段中任何一點的擾亂方向都固定為線段的垂直法向量。如圖3.6。

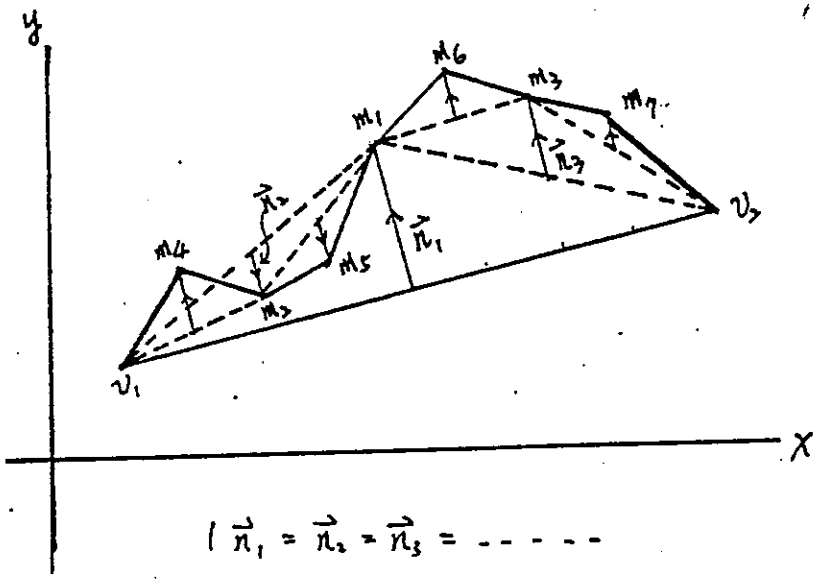


圖3.6 固定向量的擾亂

此方式對於模擬股市漲跌是非常貼切的，但對海岸線而言，規則方向的擾亂並無法真實地表現曲折的海岸線，例如內彎的情況便無法由此法模擬。

(2)現行法向量(current normal vector)

各層次(level)的分割過程中，任何一線段中點的擾亂方向為目前此線段的法向量。如圖3.7。

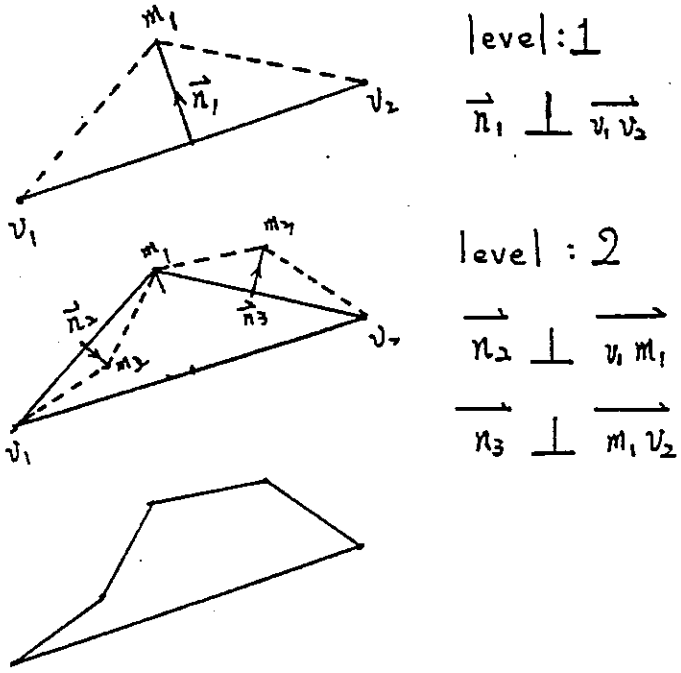


圖3.7 現行法向量的擾亂

(3)擾亂向量加現行法向量(perturbating vector+current normal vector)

以上述二種方法擾亂所得的碎形曲線都有一共同的缺點—左右比例平均。此乃因分割方法是以二分法為基礎，致使擾亂後的碎形曲線仍呈規則的分佈。為了擾亂這種規則的分佈，加入擾亂向量來擾亂現行法向量

的規則性。如下圖3.8。

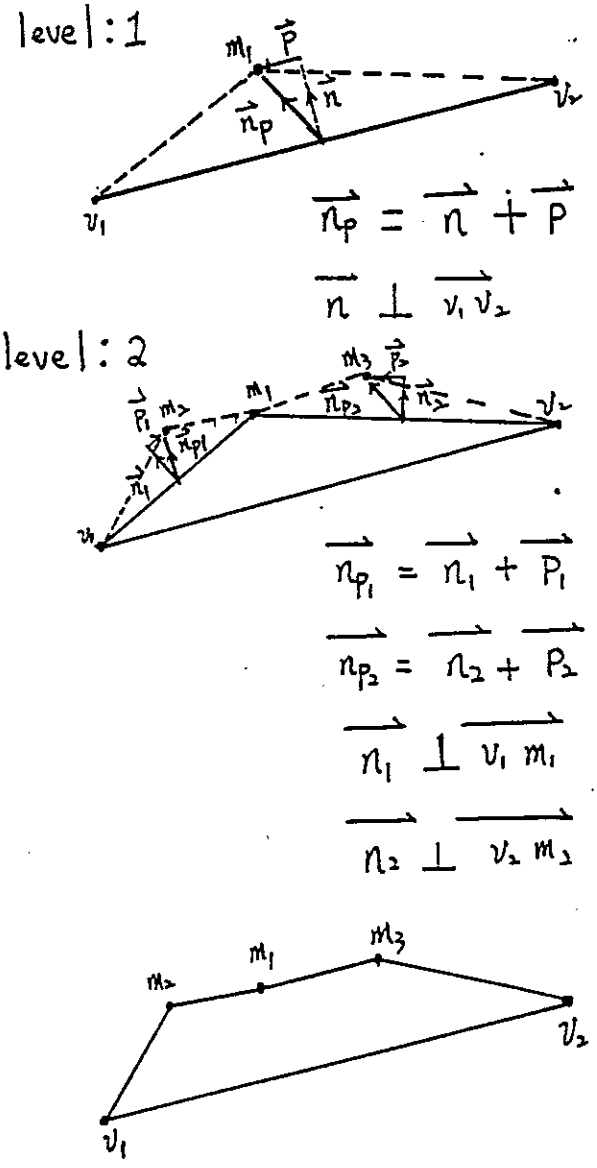


圖3.8 擾亂現行法向量的擾亂

(4)以上這三個方法都是以線段的垂直法向量為中心向量(central vector)。對於內部一致性(Internal consistency)的考慮相當合理，但是對於外部一致性(External consistency)則不合理。如下圖3.9所示：

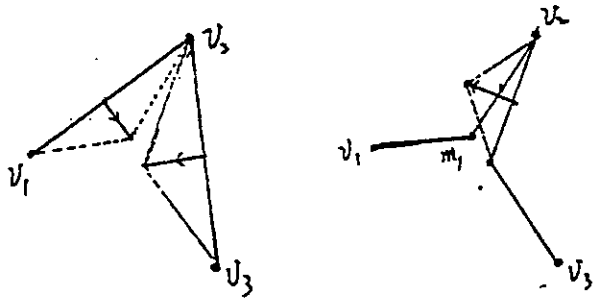


圖3.9 線段V_1 V_2 及V_2 V_3 ，以固定法向量來擾亂

發生錯誤的原因在於整體中相連的線段有關連性，是以擾亂某線段時需考慮前後相連的線段。針對此考慮先求出各線段的法向量，再用相連二線段的法向量平均求得共用點的法向量，最後以兩端點法向量$\overrightarrow{n_{V_i}}$ 及 $\overrightarrow{n_{V_{i-1}}}$ 的平均向量$\vec{n}$為中心向量(central vector)替代線段的垂直法向量，則外部一致性的缺失便迎刃而解。端點平均向量的求法如下圖3.10。

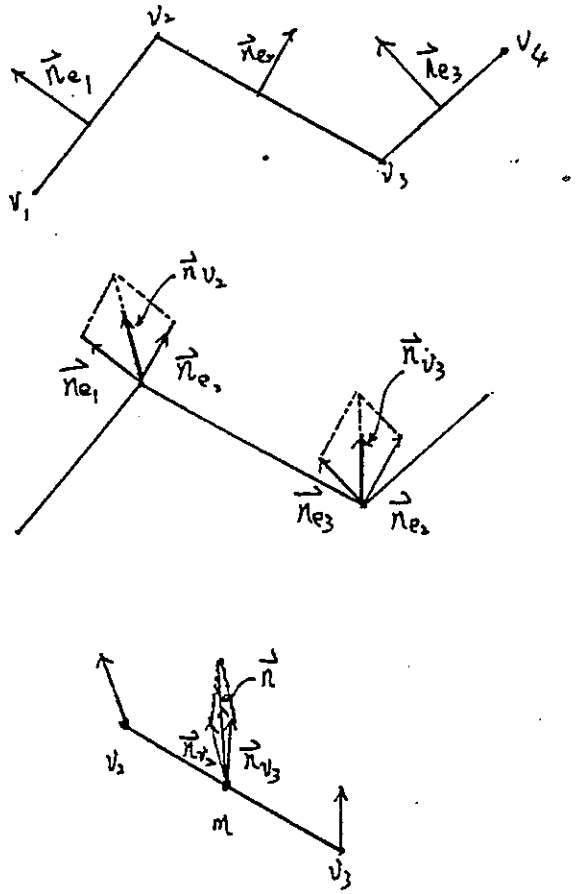


圖3.10 兩端點平均向量的求法

以上方法均只對線段(polyline)做碎形化處理，產生一連串曲折的海岸線，但原先包含此線段的多邊形平面(polygon)就變凹多邊形(concave polylgon)，如下圖3.11。

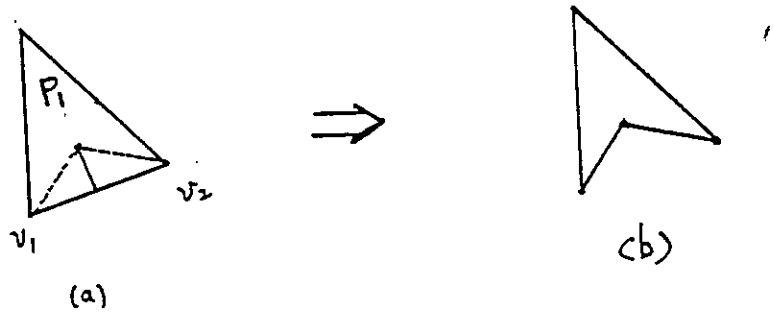


圖3.11

因為動畫者系統是以掃描線演算法(scan-line algorithm)處理多邊形平面。故輸入格式便要求輸入凸的多邊形平面(convex polygon)，其原因乃是掃描線演算法處理凹多邊形平面會發生圖3.12的錯誤。

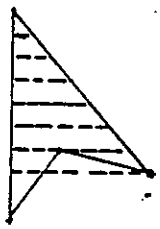


圖3.12 掃描線演算法處理時的錯誤

為了滿足動畫者的需求，在碎形化的過程中，隨機中點的移動有二種情形：

一、位移量為正，可由原三角形平面產生四個三角形平面

，見下圖3.13。

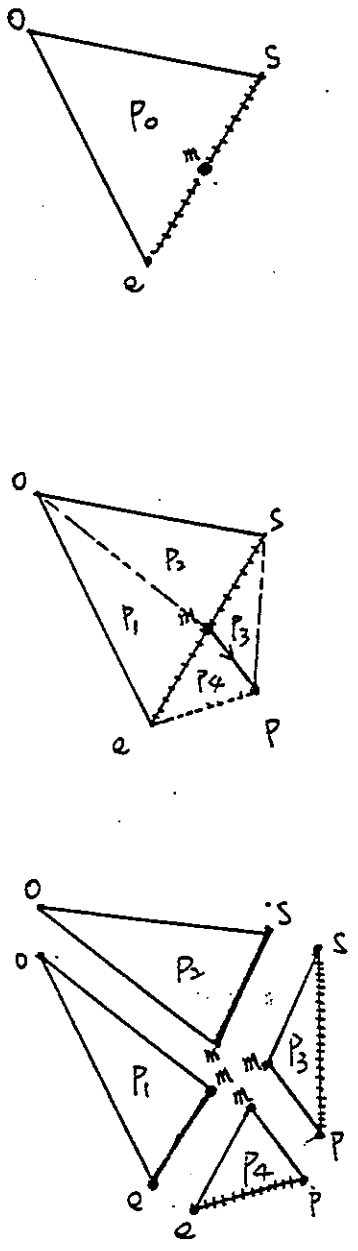


圖3.13 位移量為正時的平面切割過程

二、位移量為負時，可由原三角形平面產生二個三角形平面，如下圖3.14。

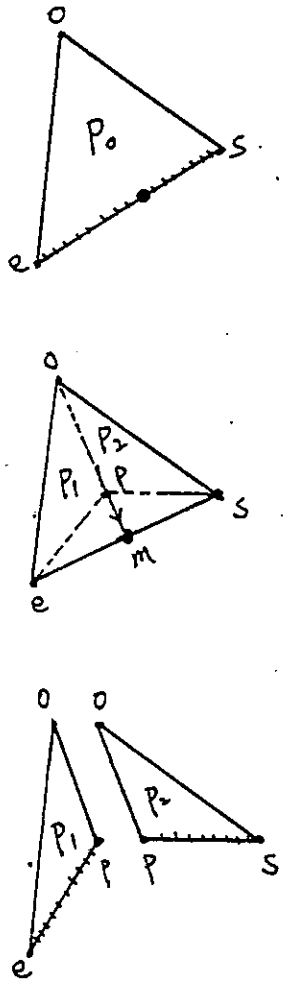


圖3.14 位移量為負時的平面切割過程

3.2.2 二維碎形處理—三角平面(triangle polygon)

前節3.2.1所述之三種方式均可推廣到二維碎形處理，只要將三角形的三邊同時做中點替換，並依圖1.5的分割方式便可。同理，在外部一致性的考慮下，又有幾種狀況需要討論：

(1)以平面法向量為中心向量：兩相連的平面經碎形化後在共用邊上會產生裂縫。如下圖3.15

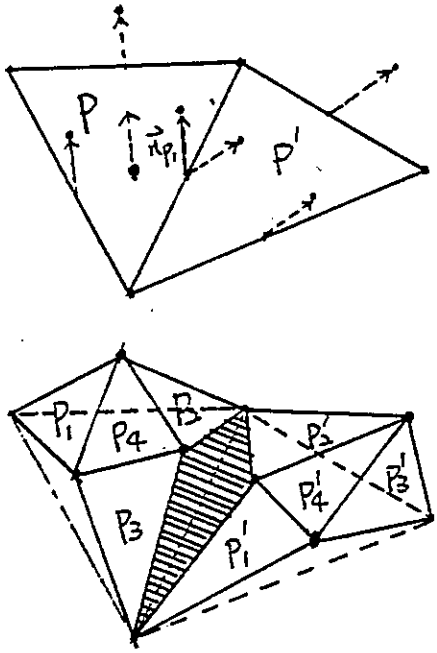


圖3.15 共用邊產生裂縫的情形

(2)以共用邊法向量為中心向量：為了解決裂縫的問題，可將相連兩平面的法向量平均成為此共用邊的法向量，使得碎形所得之結果是連續的。如下圖3.16。

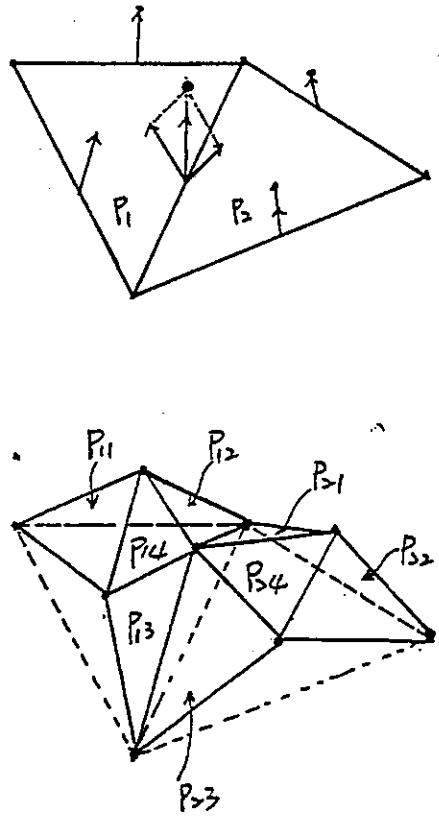


圖3.16 解決共用邊的裂縫

雖然這方法能解決裂縫。但是3.1.1節所討論的外部一致性的問題尚未解決。因為一頂點被許多邊所共用：而各邊均各自碎形化，則圖3.9的錯誤(線段可推廣到平面)亦會同樣地發生。因此下面(3)的方法就可解決這個缺點。

(3)以端點平均向量為中心向量：同3.2.1節的(4)法，(a)先求各平面法向量，(b)針對頂點求出以所有通過此頂

點的平面向量之平均為此頂點之法向量，(c)依兩端點法向量求出平均向量，即為隨機中點的移動向量。其過程如下圖3.17。

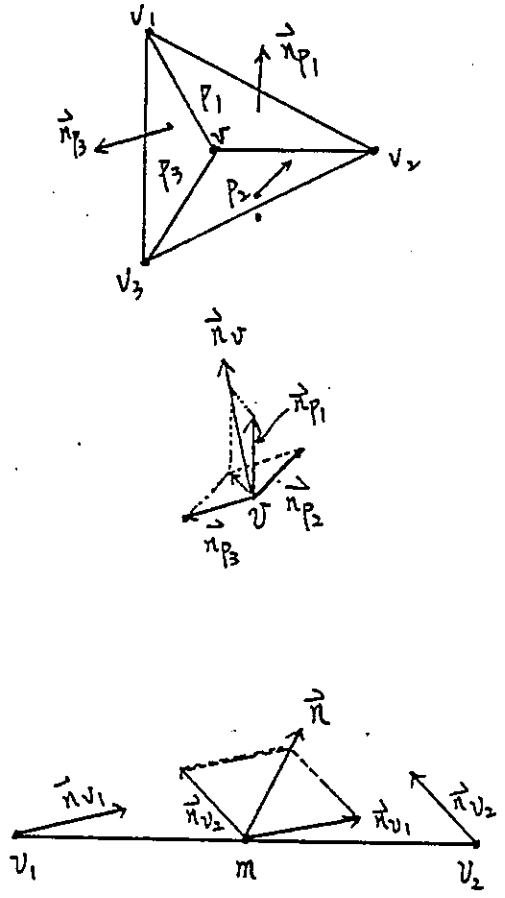


圖3.17

一般二維碎形法在處理三角形平面均以圖1.5的方式為標準。而其碎形化細分程度由階層(level)決定，而本論文則以邊長來控制碎形化細分程度，所以必須有以下的

考慮：

(一)三邊長度均小於Epsilon：

不作分割，存入已處理的平面串列中。

(二)二邊長度小於Epsilon，另一邊長度大於Epsilon：

則對大邊做分割，情形如下圖3.18。

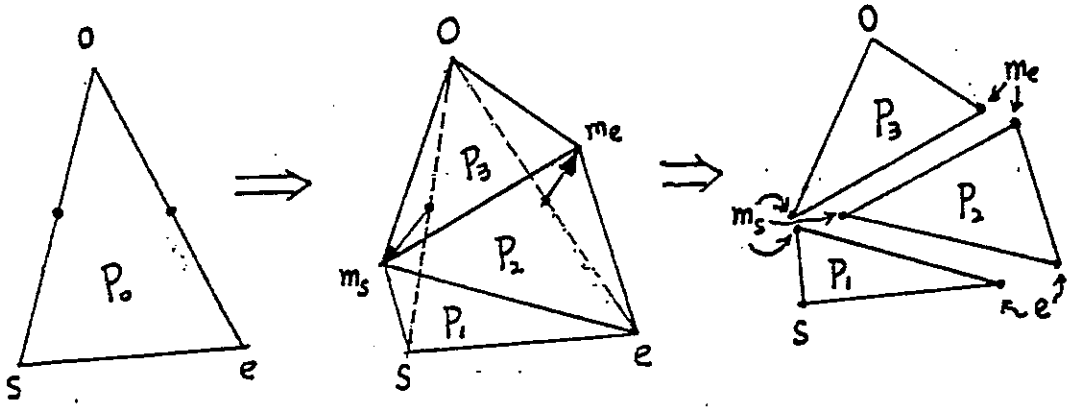


圖3.18 一三角平面分割成二個三角形平面

(三)二邊長度大於Epsilon，一邊長度小於Epsilon：

則對二個大邊做隨機中點替換，如下圖3.19。

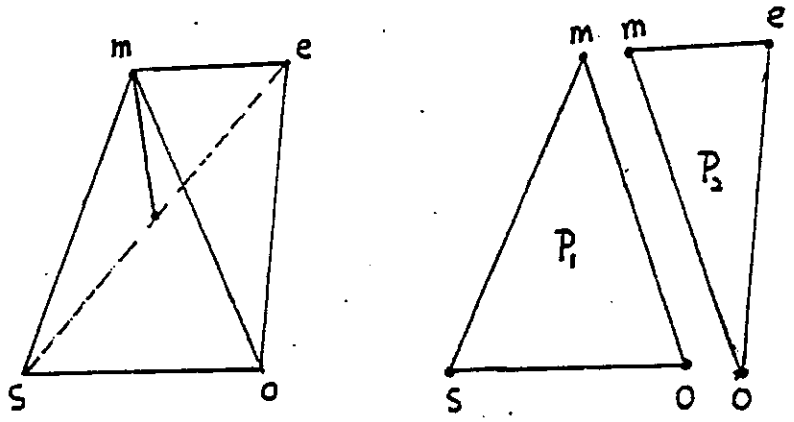


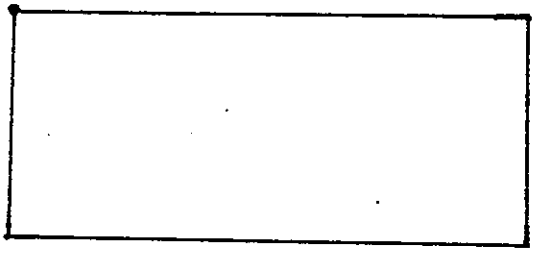
圖3.19 三角平面分割成四個三角形

(四)三邊均大於Epsilon：

分割方式同圖1.5之法。

第四章　實驗結果

依第三章所討論的方法，分別針對一維的臺灣海岸線及二維的臺灣地形做碎形處理，達到模塑碎形自然的目的。以下所得之實驗結果除了一維的臺灣海岸線及二維的臺灣地形外，本論文亦將二維碎形處理法應用到規則物體的擾，如平滑的球，經擾亂後得到凹凸不平的球形物。其他尚有橢球及甜甜圓。以下各結果的實驗描述格式如下



object (物體)

primitive (元件)

central vector (中心向量)

method (方法)

fractal parameter (碎形參數):

(1) H (亂度指標)

(2) sigma (分散值)

(3) epsilon (細分程度)

(4) fixvl (移動因子)

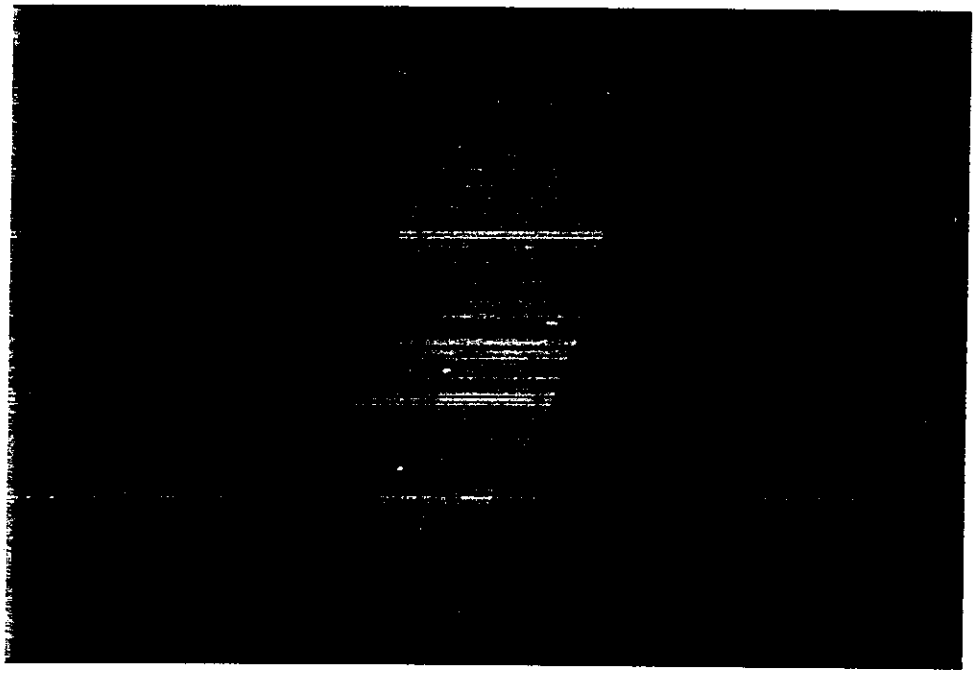


FIGURE : 1

OBJECT : Source of Taiwan

PRIMITIVE : One-dimensional coastlines

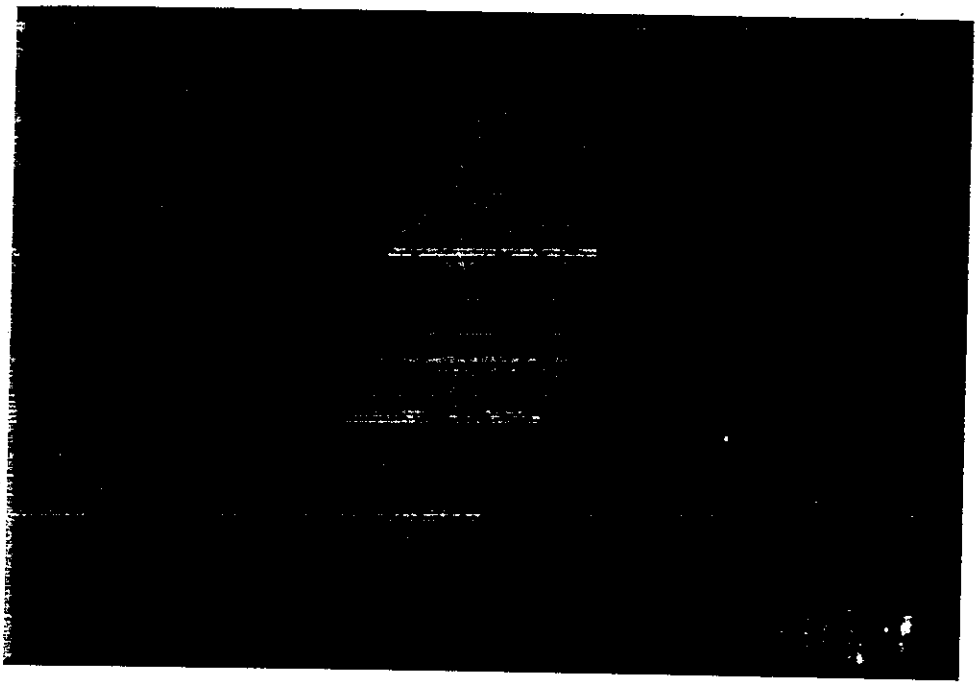


FIGURE : 2

OBJECT : Taiwan

PRIMITIVE : One-dimensional coastlines

FRACTAL METHOD : 1_D midpoint displacement

CENTRAL VECTOR : Edge-normal

NORMAL VECTOR TYPE : Fixed edge-normal

FRACTAL PARAMETER :

(1) H = 0.7

(2) Sigma = 1.0

(3) Epsilon = 0.01

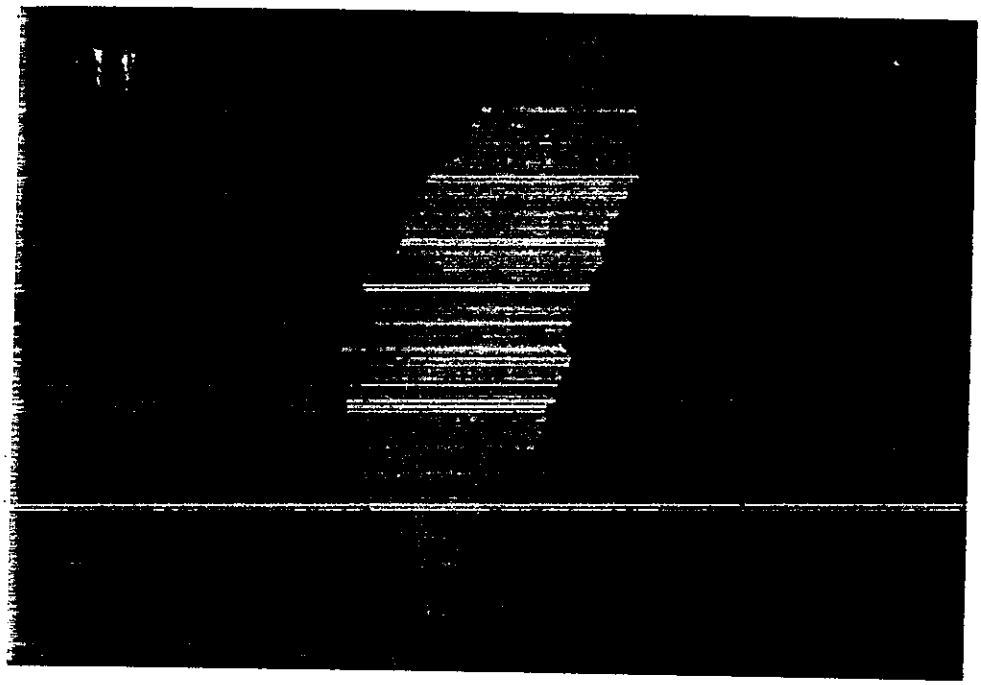


FIGURE : 3

OBJECT : Taiwan

PRIMITIVE : One-dimensional coastlines

FRACTAL METHOD : 1_D midpoint displacement

CENTRAL VECTOR : Edge-normal

NORMAL VECTOR TYPE : Fixed edge-normal

FRACTAL PARAMETER :

(1) H = 0.3

(2) Sigma = 1.0

(3) Epsilon = 0.01

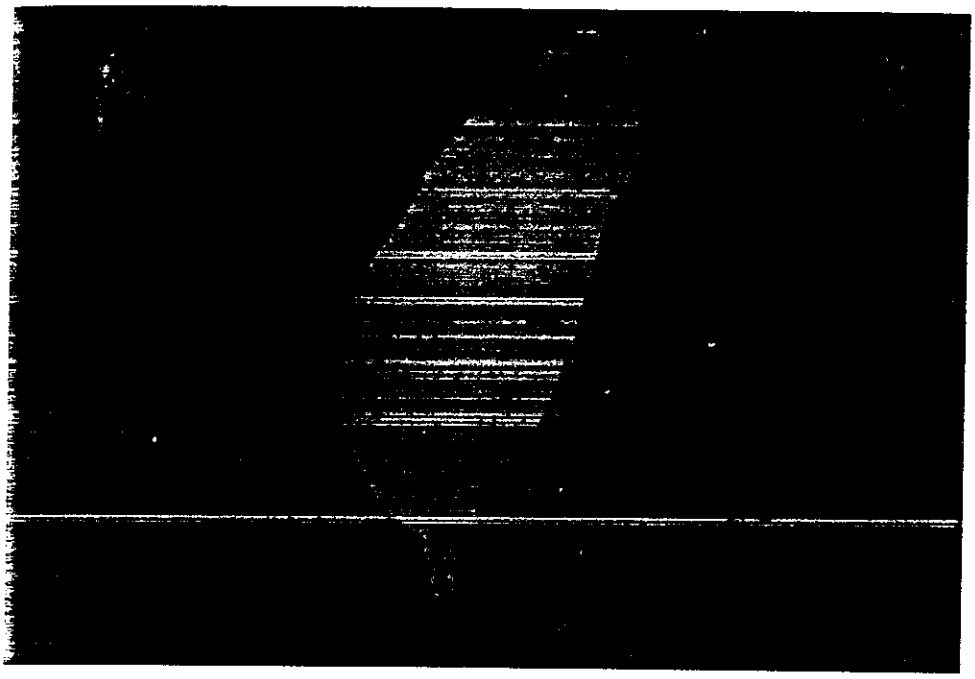


FIGURE : 4

OBJECT : Taiwan

PRIMITIVE : One-dimensional coastlines

FRACTAL METHOD : 1_D midpoint displacement

CENTRAL VECTOR : Edge-normal

NORMAL VECTOR TYPE : Current edge-normal

FRACTAL PARAMETER :

(1) H = 0.7

(2) Sigma = 1.0

(3) Epsilon = 0.01

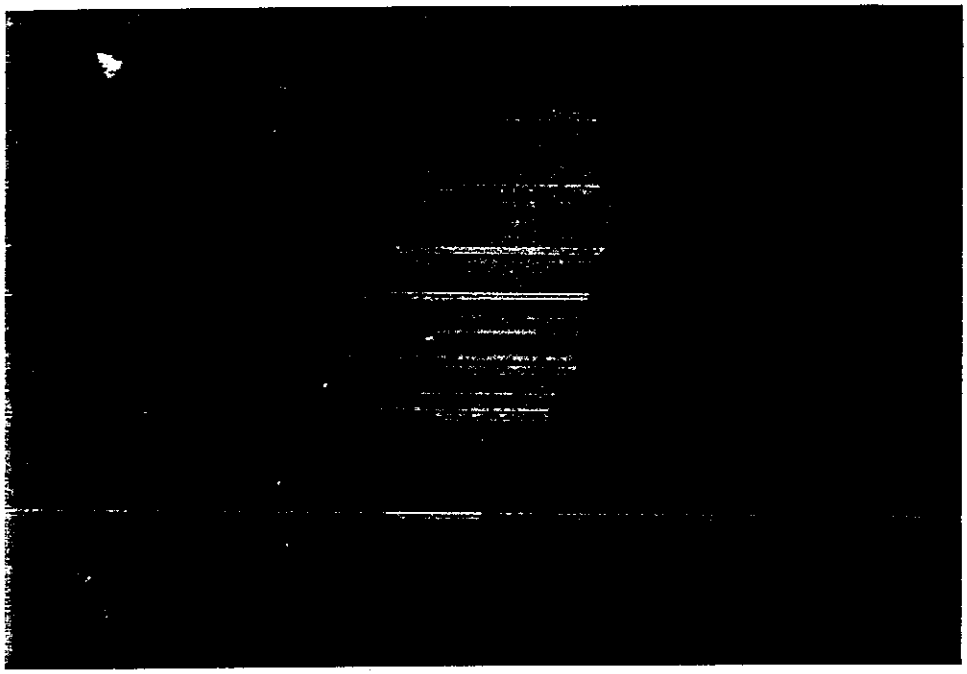


FIGURE : 5

OBJECT : Taiwan

PRIMITIVE : One-dimensional coastlines

FRACTAL METHOD : 1_D midpoint displacement

CENTRAL VECTOR : Edge-normal

NORMAL VECTOR TYPE : Current edge-normal

FRACTAL PARAMETER :

(1) H = 0.3

(2) Sigma = 1.0

(3) Epsilon = 0.01

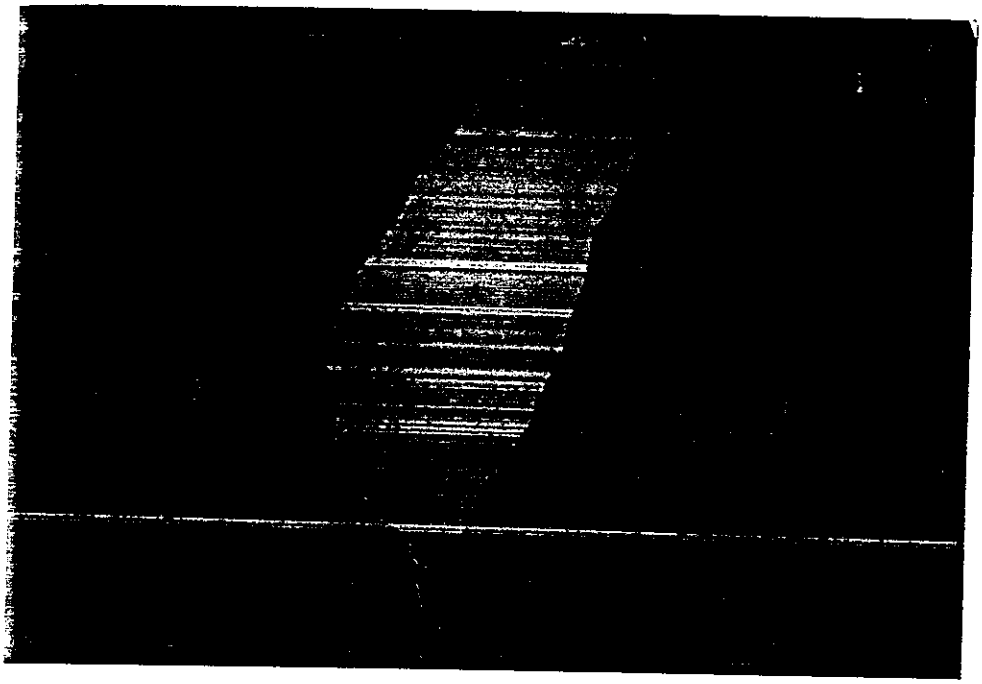


FIGURE : 6

OBJECT : Taiwan

PRIMITIVE : One-dimensional coastlines

FRACTAL METHOD : 1_D midpoint displacement

CENTRAL VECTOR : Edge-normal

NORMAL VECTOR TYPE : perturbating edge-normal

FRACTAL PARAMETER :

(1) H = 0.7

(2) Sigma = 1.0

(3) Epsilon = 0.01

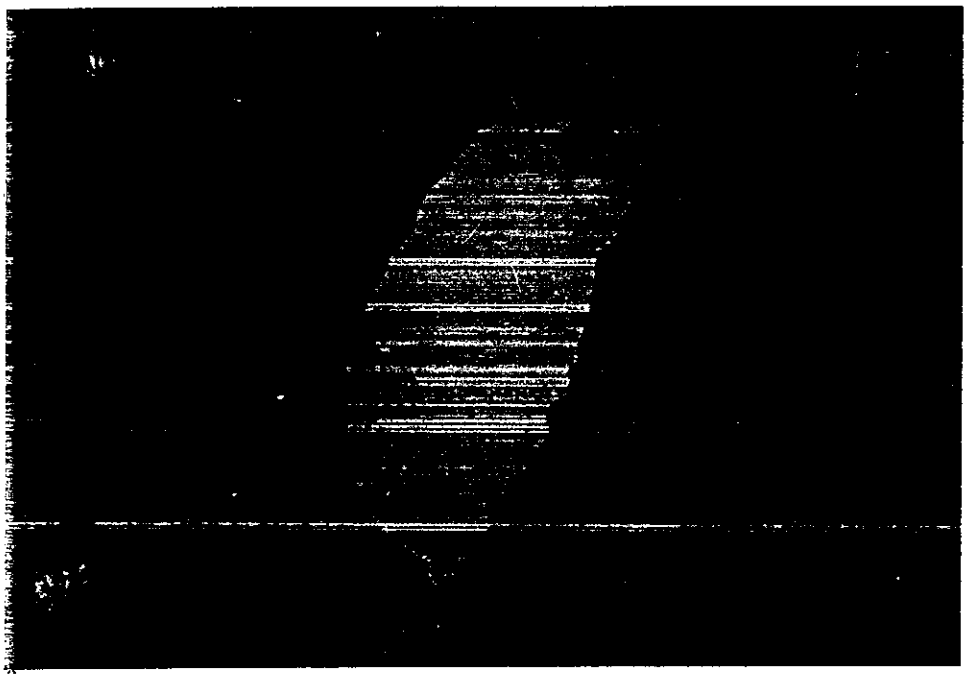


FIGURE : 7

OBJECT : Taiwan

PRIMITIVE : One-dimensional coastlines

FRACTAL METHOD : 1_D midpoint displacement

CENTRAL VECTOR : Edge-normal

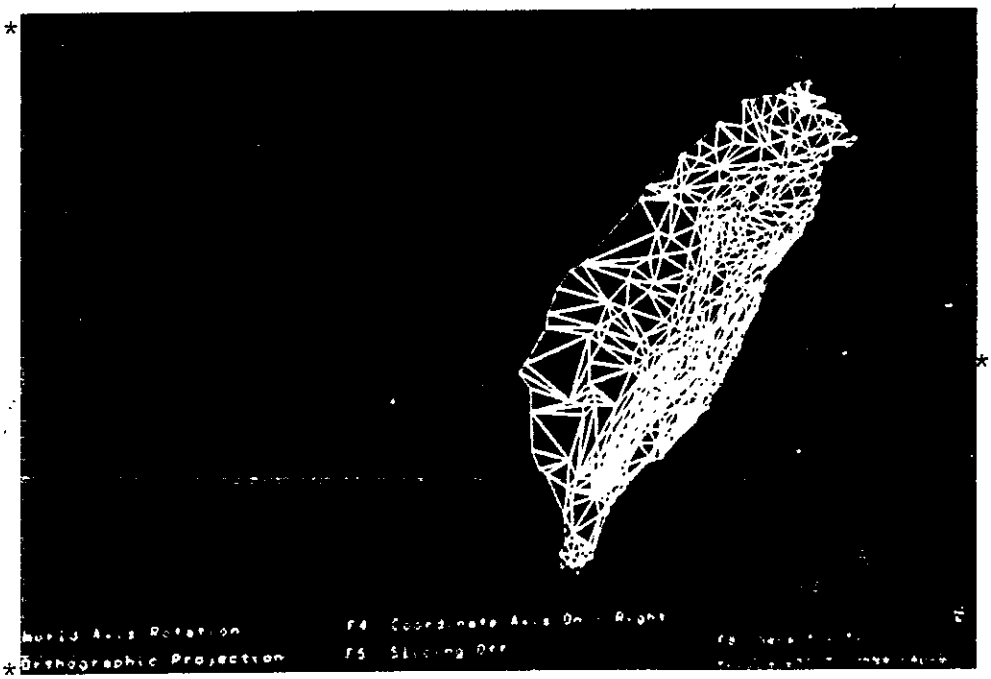
NORMAL VECTOR TYPE : perturbating edge-normal

FRACTAL PARAMETER :

(1) H = 0.3

(2) Sigma = 1.0

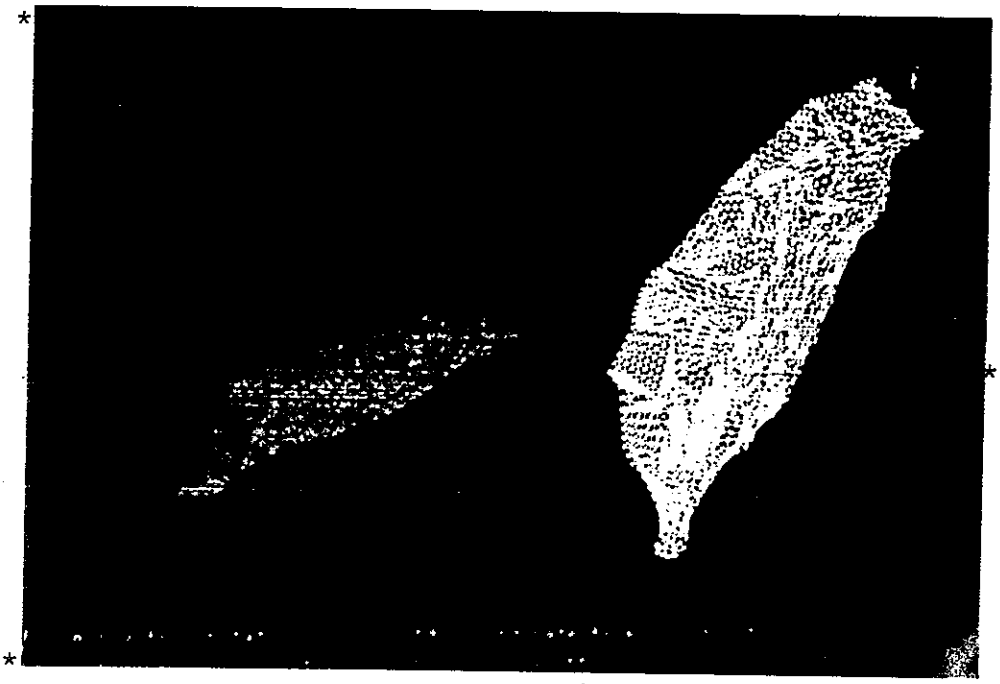
(3) Epsilon = 0.01



PICTURE : 8

OBJECT : Taiwan

PRIMITIVE : Two-dimensional mountain-surface



PICTURE : 9

OBJECT : Taiwan

PRIMITIVE : Two-dimensional mountain-surface

FRACTAL METHOD : Two-dimensional midpoint displacement

CENTRAL VECTOR : Average-vtx-normal

NORMAL VECTOR TYPE : Current-Average-vtx-normal

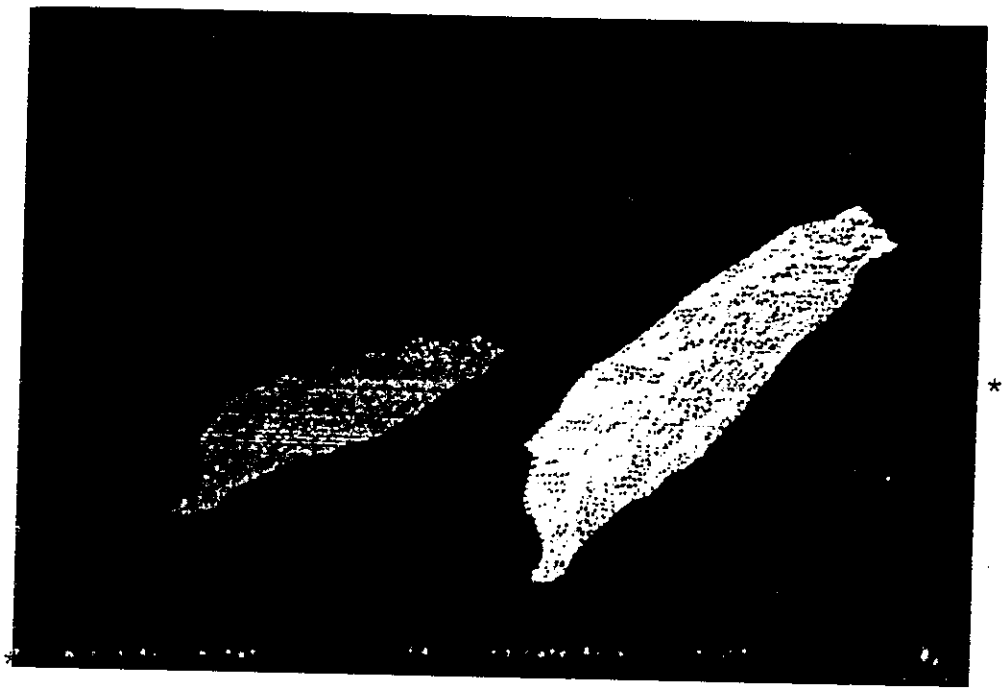
FRACTAL PARAMETER :

(1) H = 0.3

(2) Sigma = 0.5

(3) Epsilon = 0.1

(4) Fixvl = 0.4



PICTURE : 10

OBJECT : Taiwan

PRIMITIVE : Two-dimensional mountain-surface

FRACTAL METHOD : Two-dimensional midpoint displacement

CENTRAL VECTOR : Average-vtx-normal

NORMAL VECTOR TYPE : Current-Average-vtx-normal

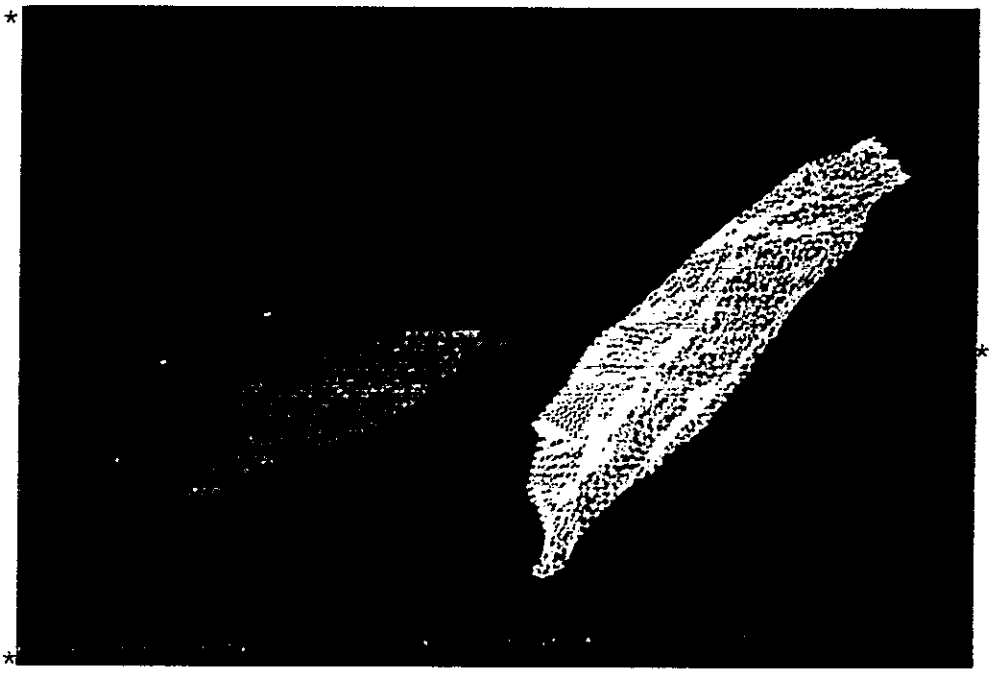
FRACTAL PARAMETER :

(1) H = 0.7

(2) Sigma = 0.5

(3) Epsilon = 0.1

(4) Fixvl = 0.4



PICTURE : 11

OBJECT : Taiwan

PRIMITIVE : Two-dimensional mountain-surface

FRACTAL METHOD : Two-dimensional midpoint displacement

CENTRAL VECTOR : Average-vtx-normal

NORMAL VECTOR TYPE : Perturbating + Current-A-vtx-normal

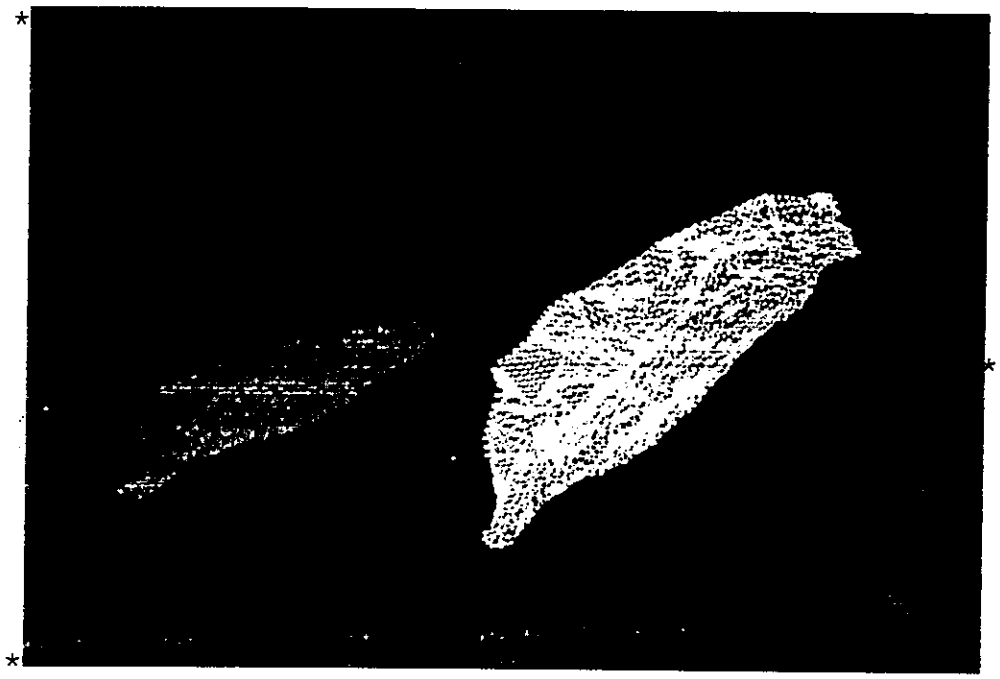
FRACTAL PARAMETER :

(1) H = 0.3

(2) Sigma = 0.5

(3) Epsilon = 0.1

(4) Fixvl = 0.4



PICTURE : 12

OBJECT : Taiwan

PRIMITIVE : Two-dimensional mountain-surface

FRACTAL METHOD : Two-dimensional midpoint displacement

CENTRAL VECTOR : Average-vtx-normal

NORMAL VECTOR TYPE : Perturbating + Current-A-vtx-normal

FRACTAL PARAMETER :

(1) H = 0.7

(2) Sigma = 0.5

(3) Epsilon = 0.1

(4) Fixvl = 0.4

第五章　結論

自然界中，萬物各異，呈現的碎形現象各有特色，不同的物體，適合不同的表現方式，因此自然界中的碎形物體模擬尚有許多需要鑽研的領域，像閃電，天空中的星星，雲霧，分子運動軌跡及各類草樹等等的模擬，在動畫的製作考慮上都能增加圖形的真實效果，提高動畫的品質。但是在模擬的過程中，常常引用到複雜的數學或物理學的理論，對使用者而言較難理解，故本論文研究其數學及物理學理論的實際意義，將理論現象實際化，針對不同的物體採用不同組合的方法處理，當然各物體亦會有相同之處，使得物體的碎形處理關係複雜，因此本論文希望建立一種物體導向語言(object-oriented language)為主的介面架構，解決物體關複雜的關係，然而本論文無對各種自然物逐一作探討，僅詳細地討論了一維及二維碎形處理法(採用隨機中點替換法)在海岸線及地形上碎形化的效果，提供後繼研究者一些參考資訊。在此我們不但希望對研究工作者有些貢獻，並希望能應用到廣泛的廣告、影的製作及軍事訓練甚至現象的模擬，對改善人類生活及幸福有一點點幫助。

参考文献

[1] Mandelbrot B, van Ness J (1968) Fractional Fractional brownian motions, fractional noise and applications. SIAM Review 10(4):422-437

[2] Mandelbrot B (1977) Fractal:form, chance and dimension, WH Freeman, San Francisc0

[3] Mandelbrot B(1982) The fractal geometry of nature. WH Freeman, San Francisco

[4] Fournier A, Fussell D, Carpenter L (1982) Computer rendering of stochastic models. comm. ACM 25(6): 371-384

[5] Miller GSP (1986) The definition and rendering of terrian maps. Proc. SIGGRAPH'86 Computer Grapics 20(4):39-48

[6] Norton A (1982). Generation and display of geometric froctal in 3_D. Proc. SIGGRAPH'86 Computer Grophics 16(3):61-67

[7] Peitgen H.O.,Saupe D (1988) The Science of fractal images. Springer-Verlag New York

[8] Perrin J (1909) Mouvement brownien et r'ealite' mol'eculaire. Annales de chimie et de physique VILL 18, 5-114. Trans. F. Soddy, as Brownian Movement and Molecular Reality. London: Taylor & Francis.

[9] Perrin J. 1913. Les Atomes. Paris: Alcan. A 1970 reprint by Gallimard supersedes several revisions that had aged less successfully, English translation:Atoms, by D.L. Hammick; London: Constable. New York: Van Nostrand. Aiso translated into German, Polish, Russian, Serbian and Japanese.

[10] Thompson D'A. W. 1917-1942-1961. On growth and form. Cambridge University Press. The dates refer to the first, second and abridged edition.

[11] Mood A.M., Graybill F.A., Boes D.C. (1974) Introduction to the Theory of statistics. McGraw-Hill Book Company

Appendix ： 程式說明

依照論文中第三章第二節的討論及分類，吾人分別將各方式實際寫成程式，以下程式能將粗略物體的平面檔案碎形化成細碎密緻的物體外形。以下便依第三章第二節的分類順序說明對應程式的內容及流程：

一、一維碎形處理 (/usr/local/graphics/ps300/student/skc/plpoly)

(一) 中心向量為線段的垂直向量 (normal vector of polylines)

(1) 固定法向量 —— /usr/local/graphics/ps300/student/skc/plpoly/fnewld.c

程式開頭描述本程式的架構及各副程式的關係。而各副程式的功能如下：

main()：主程式，讀入碎形化控制參數，並啟動各副程式。

pline()：將描述物體形狀的平面檔讀入，並記錄；物體的每段線段。

mpbm()：取出欲處理的線段 (polyline) 並呼叫隨

机中央處理程式 mprecurs() 。

initgauss()：設定 gauss 分佈的亂拔產生常式所用到的常數。使得每一線段的一連串亂數均相同。

gauss()：產生符合 gauss 分佈的亂數，以利 mprecurs() 產生零亂效果。

mprecurs()：以遞迴方式計算線段中碎形化所需之隨机中央的位移量（垂方向已固定了）

normpert()：依 mprecurs() 所得之位移量，移动隨机中央。

lsrch()：linear search。檢查此 polyline 之狀態。

fraout()：輸出碎形化平面檔案。

lfind()：確線段中隨机中央的存放順序。

輸入的物體角圖片 1 (/usr/local/SUN/skc/pledge/stai.ras)

產生的物體 " 2 及 3 (/usr/local/SUN/skc/pledge/fstai2.ras 及 /fstai3.ras)

```
/* purpose -- make a midpoint displacement method use a
     _         gauss noise function control by level

 |main() -- main procedure read control parameters
 |           from *.par
 |
 |--|pline() -- read points and polylines from polygon
 |               file *.std
 |
 |--|mpbm() -- random midpoint displacement simulate
 |   |          Brownian motion
 |   |
 |   |---|gauss() -- random number generator of Gaussian
 |   |                distribution
 |   |
 |   |---|initgauss() -- set contants value of gauss
 |   |                   routine
 |   |
 |   |---|mprecurs() --| calculating random midpoints by
 |       |             | recurrsion
 |       |--|mprecurs()|
 |
 |--|normpert() -- moving random midpoints by fixed
 |   |              edge normal
 |   |
 |   |---lsrch() -- check statu of polylines
 |
 |--|fraout() -- output result polygon file form - *.epf
    |
    |---lfind() -- check storing order of new midpoints
                   of polylines

*/

#include <stdio.h>
#include <math.h>

#define sqr(x)  ((x)*(x))
#define POWER      31.
#define MAXLEVEL   16
#define Nrand       5
#define MAXPTS  16384
#define MAXTAB   1024
#define MAXOFF   1024
#define MAXPOLY     2
#define MAXVTX   8192
#define T           1
#define F           0

int       seed,nrand,maxlevel;
int       npts,npolys,nlines,ltabc;
float     sigma,h;
float     epsilon,scale,eyedist,movefac;
float     arand,gadd,gfac;

int     polyarr[ MAXPOLY ][ MAXVTX ];
```

```
57                                          /* polyline array */
58  struct   { int   ind;
59             int   svtx;
60             int   evtx;  }  addvtxs[ MAXTAB ];
61                   /* looking table of every pair vtxs */
62  struct  vlocat { float    x;
63                   float    y;
64                   float    z; }
65          varr[ MAXPTS ];
66                              /* result vertice array  */
67  float   offset[ MAXOFF ];
68                             /* offset of each vertice */
69
70  main(argc,argv)
71    int   argc;
72    char *argv[];
73    {  int   i,j,n;
74       char  ch[3];
75       int   vtx1,vtx2,maxdiv;
76       float dist,sqrx,sqry;
77      FILE    *input;
78      char    filename[512];
79      char    string[512];
80
81      pline(argv);
82       ltabc = 0;
83
84  /*****---------- READ PARAMETERS -----------------*****/
85      strcpy(filename,argv[1]);
86      strcat(filename,".par");    /*  open *.par file  */
87     input = fopen(filename,"r");
88
89                  /*  read until EOF or "data" keyword  */
90      do {fgets(string,512,input);}
91      while ((*string != NULL)
92              && (strncmp(string,"parameters",10) != 0));
93
94      fgets(string,512,input);
95       sscanf(string,"%f",&scale);  /*refined level of object*/
96
97      fgets(string,512,input);
98       sscanf(string,"%f",&eyedist);/*distance of eyes to object*/
99       movefac = eyedist * 0.01;
100                              /** scalefac = 0.1 * scale;
101                                 midmove = movefac * scalefac;*/
102      epsilon = movefac / scale;
103                                /*** error distance of eyes ***/
104 err1:
105     fgets(string,512,input);
106      sscanf(string,"%d",&seed);
107                    /*initial value of randam number generator*/
108     fgets(string,512,input);
109      sscanf(string,"%f",&h);
110                                /*fractal h-dimension of object*/
111     fgets(string,512,input);
112      sscanf(string,"%f",&sigma);/*sigma of noise distribution*/
```

```
113       if (h>1. || h<0.) {printf("h == [0,1] reenter!! \n");
114                          goto err1;}
115     fclose(input);
116
117   /*****---------- END OF READ PARAMETERS ----------*****/
118
119      for(i = 0;i < npolys;i++) {
120        printf(" polygon  i = %d\n",i); /* no. of polygons */
121        nlines = polyarr[i][0];
122
123       for(j = 1;j <= nlines;j++) {      /* no. of polyline */
124        printf("parameters of polyline j = %d\n",j);
125        vtx1 = polyarr[i][j];
126        if (j == nlines) vtx2 = polyarr[i][1];
127           else  vtx2 = polyarr[i][j+1];
128        n = 1;maxlevel = 0;
129        sqrx=varr[vtx2].x-varr[vtx1].x;
130        sqry=varr[vtx2].y-varr[vtx1].y;
131        dist=sqrt(sqr(sqrx)+sqr(sqry));
132        maxdiv = (int)(dist/epsilon) + 1;
133                              /* compute no of divided seg. */
134        do {n = 2 * n;maxlevel++;} while(n <= maxdiv);
135        polyarr[i][0] = polyarr[i][0] + n -1;
136
137      mpbm(n); /*** MidPointBMotion routine ***/
138
139      normpert(n,i,j); /*** normal vector perturbation ***/
140
141    no:printf("\n");
142        }
143      }
144
145      fraout(argv);/* output result "epsilon" datafile *.epf */
146
147     }
148   /**---- read point and polygon suport midpoint routine --**/
149
150   pline(str)
151    char  *str[];
152   {
153       FILE    *input;
154       char    filename[512];
155       int     i,a,b;
156       char    string[512],str2[512];
157
158       strcpy(filename,str[1]);
159       strcat(filename,".std");  /*  open *.std file  */
160      input = fopen(filename,"r");
161
162                                 /*read until EOF or "data" keyword*/
163       do {fgets(string,512,input);}
164       while ((*string != NULL)
165             && (strncmp(string,"data",4) != 0));
166       sscanf(string, "%s %d %d",str2,&a,&b);
167       npts = (short) a + 1;
168       npolys = (short) b;
```

```
169
170       for (i = 1;i < npts;i++)  /* read in and copy out points */
171         { fgets(string,512,input);
172           sscanf(string,"%f %f %f",&varr[i].x,&varr[i].y,&varr[i].z);
173         }
174
175       for (i = 0;i < npolys;i++)                    /* do polygons */
176         { short j,k,l;
177           fgets(string,512,input);
178
179           j = 0;
180           while(string[j] == ' ') j++;          /* ignore blanks */
181           l = polyarr[i][0] = atoi(&string[j]);
182
183           for (k = 1; k <= polyarr[i][0]; k++)
184           { while(string[j] != ' ') j++;
185             polyarr[i][k] = atoi(&string[j]);
186             while(string[j] == ' ') j++;/*read vtxs of polyline*/
187           }
188           polyarr[i][l+1] = -1;
189         }
190    fclose(input);
191  }
192
193  /***-----mpbm ( offsetXarr , maxlevel , sigma , seed );--***/
194
195  mpbm(n)
196   int n;
197   {
198      int       i;
199      float     gauss();
200      float     delta[ MAXLEVEL ];
201
202      initgauss(seed);
203
204                                /** Variance of every level **/
205      for (i = 1;i <= maxlevel;i++)
206          delta[ i ] = sigma * pow(0.5,((float)i)*h)
207                             * sqrt(1 - pow(2.,2.*h - 2.));
208      offset[0] = sigma * gauss();
209      offset[n] = sigma * gauss();
210                      /** Calculus offset of new midpoints **/
211          mprecurs(0,n,1,delta);
212      offset[0] = 0.;
213      offset[n] = 0.;
214
215   }
216
217  /***---midpointrecursion (index0,index2,level,delta )---***/
218
219  mprecurs(index0,index2,level,delta)
220    int   index0,index2,level;
221    float delta[ MAXLEVEL ];
222    { int   index1;
223
224      index1 = (index0 + index2)/2;
                                /** offset of midpoint **/
```

```
225       offset[index1] = 0.5 * (offset[index0] + offset[index2])
226                                + delta[level] * gauss() ;
227        if (level < maxlevel)
228          {
229            mprecurs(index0,index1,level+1,delta);
230            mprecurs(index1,index2,level+1,delta);
231          }
232     }
233
234   /**-INITGAUSS routine-set initial value of gauss routine-**/
235
236    initgauss(seed)
237      int seed;
238      {
239       arand = pow(2.,POWER) - 1.;
240       gfac = sqrt(12./ Nrand) / arand;
241       srand(seed);
242      }
243
244   /**-GAUSS routine-gauss randam number generator----------**/
245
246   float  gauss()
247      {
248       float  sum;
249       int    i;
250
251       sum = 0.;
252       for (i = 1;i <= Nrand;i++)
253           sum = sum + (rand() - arand/2);
254       return(gfac * sum);
255                                /* return(fabs(gfac * sum)); */
256      }
257   /* end of gauss routine ***/
258
259   /***-------NORMAL PERTURBATION( offset [ 1 to n-1 ] )---***/
260
261   normpert(n,l,k)
262     int   n,l,k;
263     {
264       int   vtx1,vtx2,i,ind;
265       short sign;
266       float delx,dely;
267
268      vtx1 = polyarr[l][k];
269      if (k == nlines) vtx2 = polyarr[l][1];
270         else  vtx2 = polyarr[l][k+1];
271      if (vtx1 < vtx2)
272              {sign = -1;ind = vtx2*(vtx2 - 1)/2 + vtx1;}
273         else {sign = 1;ind = vtx1*(vtx1 - 1)/2 + vtx2;}
274                               /**check ind in lookup table**/
275      if (lsrch(ltabc,ind) == T) return;
276                             /* hold start vtx of vtxs sequence */
277       addvtxs[ ltabc ].ind = ind;
278       addvtxs[ ltabc ].svtx = npts;
279                                        /**   delta of offset   **/
280       delx = (varr[vtx2].x - varr[vtx1].x ) /(float) n;
```

```
281     dely = (varr[vtx2].y - varr[vtx1].y ) /(float) n;
282                                     /**   move by offset    **/
283     if (sign == -1)
284        for(i = n - 1;i > 0;i--){ /* reverse new vtxs seq. */
285           varr[npts].x = varr[vtx1].x + delx*i - dely*offset[i];
286           varr[npts].y = varr[vtx1].y + dely*i + delx*offset[i];
287           npts = npts + 1;
288        }
289      else for(i = 1;i < n;i++){     /* acoording to vtxs seq. */
290           varr[npts].x = varr[vtx1].x + delx*i - dely*offset[i];
291           varr[npts].y = varr[vtx1].y + dely*i + delx*offset[i];
292           npts = npts + 1;
293           }
294                                  /* hold end vtx of vtxs sequence */
295     addvtxs[ ltabc ].evtx = npts - 1;
296     ltabc = ltabc + 1;
297   }
298   /**----------------- end of normperturbation  ------------***/
299
300   /**-------------- Linear Search routine --------------------**/
301
302   lsrch(upper,key)
303    int   upper,key;
304    {   int   i;
305
306        i = 1;
307        while (i < upper) {
308           if (addvtxs[ i ].ind == key) return(T);
309           i = i + 1;
310        }
311        return(F);
312    }
313   /***--- end of Linear Search routine ---------------------***/
314
315   /***-----------RFAOUT routine ----------------------------***/
316
317   fraout(str)
318     char *str[];
319    {
320      FILE  *output;
321      int   i;
322      char  filename[512];
323
324      strcpy(filename,str[1]);
325      strcat(filename,".epf");  /*  open *.fra file  */
326      output = fopen(filename,"w");
327
328      fprintf(output,"%s %d %d \n","data",npts - 1,npolys);
329                                  /* write all points *.x *.y *.z */
330      for (i = 1;i < npts; i++)
331        fprintf(output,"%f %f %f \n",varr[i].x,varr[i].y,varr[i].z);
332
333      for (i = 0; i < npolys; i++)                  /* do polygons */
334       { short j,k,vtx1,vtx2,svtx,evtx;
335         int   sign,itab,ind;
336         fprintf(output," %d",polyarr[i][0]);
```

```
337         j = 1;
338       while( polyarr[i][j] != -1) {
339           vtx1 = polyarr[i][j];
340           fprintf(output," %d",vtx1);
341           if (polyarr[i][j+1] == -1) vtx2 = polyarr[i][1];
342              else  vtx2 = polyarr[i][j+1];
343           if (vtx1 < vtx2)
344                 {sign = -1;ind = vtx2*(vtx2 - 1)/2 + vtx1;}
345              else {sign = 1;ind = vtx1*(vtx1 - 1)/2 + vtx2;}
346          itab = lfind(ltabc,ind);
347           if (itab == -1) goto errout;
348           svtx = addvtxs[ itab ].svtx;
349           evtx = addvtxs[ itab ].evtx;
350           if (sign == -1) /** reverse order seq. of newpoint **/
351              for(k=evtx;k>=svtx;k--) fprintf(output," %d",k);
352                        /** positive order seq. of newpoint **/
353             else for(k=svtx;k<=evtx;k++) fprintf(output," %d",k);
354           j = j + 1;
355          }
356          fprintf(output," \n");
357       }
358  errout:fclose(output);
359   }
360  /**---------------- END OF FRAOUT ROUTINE ------------------**/
361
362  /**-------------- Linear find index of newpoint ------------**/
363
364   lfind(upper,key)
365     int   upper,key;
366     { int  i;
367
368       i = 0;
369       while (i < upper) {
370          if (addvtxs[ i ].ind == key) return(i);
371          i = i + 1;
372       }
373       return(-1);
374     }
375  /**------------ end of linear find index of newpoint ---------**/
```

(2) 現行法向量 —— Cnem1d.C

(3) 擾亂向量加現行法向量 —— Pnem1d.C

由於(2)及(3)相差僅在 mprecurs() 中加入擾亂向量(由 nordist() 所產生的亂數所控制)，故吾人將這二程式合併成 cpnem1d.C (/usr/local/graphics/ps300/student/skc/plpoly/cpnem1d.C)

main() : 同前。

pline() : 同前。

initgauss(): 同前。但因位置不同於 fnem1d.C，使得每一線段所得到的一串亂數均不同。

mpbm() : 同前。

mprecurs(): 同前。但每次均需重新計算現行法向量及擾亂向量。

(a) nor = 0; ⟶ cnem1d.C.

(b) nor = delta[level] * nordist(); ⟶ pnem1d.C

gauss() : 同前。

nordst()：normal distribution 和 gauss 類似，控制擾乱向量的擾乱程度。

lsrch()：同前。記錄各線段移动隨机中选的顺序。

fraout()：同前。

lfmd()：同前。

產生 图4.dat 存於 /usr/local/SUN/skc/pledge/ctai7.ras

图5 〃 〃 /ctai3.ras

图6 〃 〃 /ptai7.ras

图7 〃 〃 /ptai3.ras

```
/* purpose -- make a midpoint displacement method use a
     —          gauss  noise  function control by epsilon


 |main() --  main  procedure  read  control  parameters
 |           from  *.par
 |
 |---pline() -- read  points and polylines from polygon
 |              file *.std
 |
 |---initgauss() -- set contants value of gauss routine
 |
 |---|mpbm() -- random  midpoint displacement  simulate
 |   |          Brownian  motion
 |   |
 |   |---|mprecurs() --| calculating  random  midpoints
 |       |             | by  recurrsion
 |       |             |
 |       |---mprecurs()|
 |       |
 |       |---gauss() --  random  number  generator  of
 |       |               Gaussian  distribution
 |       |
 |       |---nordist() -- producing  random  number  of
 |                         normal    distribution    to
 |                         perturbate   current   normal
 |                         vector
 |
 |---lsrch() -- check statu of polylines
 |
 |---normpert() -- moving  random  midpoints  by  order
 |
 |---|fraout() -- output result polygon file form *.epf
     |
     |--lfind() -- check storing order of new midpoints

*/

#include <stdio.h>
#include <math.h>

#define sqr(x)   ((x)*(x))
#define POWER      31.
#define MAXLEVEL   16
#define Nrand       4
#define MAXPTS  16384
#define MAXTAB   1024
#define MAXNEW   1024
#define MAXPOLY     2
#define MAXVTX   8192
#define T           1
#define F           0

int       seed;
float     arand,gfac;
float     sigma,h;
```

```
 57  double    epsilon;
 58  float     scale,eyedist;
 59  float     epfac,scalefac,movefac;
 60  int       npts,npolys,nlines,ltabc;
 61
 62  int     polyarr[ MAXPOLY ][ MAXVTX ];          /*  polyline array  */
 63  struct  { int   ind;
 64            int   svtx;
 65            int   evtx; } addvtxs[ MAXTAB ];
 66                                        /*  looktab of all pair vtxs  */
 67  struct  vlocat { float    x;
 68                   float    y;
 69                   float    z; } ;                /** vertice type **/
 70  struct  vlocat   varr[ MAXPTS ];        /** result vertice array **/
 71
 72  struct  plocat { float    x;
 73                   float    y; } ;                /**   point type **/
 74  struct  plocat  newpts[ MAXNEW ]; /* location of every new points */
 75
 76
 77  main(argc,argv)
 78    int   argc;
 79    char *argv[];
 80    {  int   mpbm(),newvtc,sign;
 81       int   ind,i,j,vtx1,vtx2;
 82     FILE    *input;
 83     char    filename[512];
 84     char    string[512];
 85
 86     pline(argv[1]);/* read *.std and get all points & polylines */
 87      ltabc = 0;
 88
 89  /*****--------------- READ PARAMETERS ----------------------*****/
 90
 91     strcpy(filename,argv[1]);
 92     strcat(filename,".par");                  /*  open *.par file  */
 93    input = fopen(filename,"r");
 94
 95                               /*  read until EOF or "data" keyword  */
 96     do {fgets(string,512,input);}
 97     while ((*string != NULL)
 98             && (strncmp(string,"parameters",10) != 0));
 99
100     fgets(string,512,input);
101      sscanf(string,"%f",&scale);     /** refined level of object **/
102
103     fgets(string,512,input);
104      sscanf(string,"%f",&eyedist); /* distance of eyes to object */
105      epfac = eyedist * 0.01;
106                                  /**scalefac = 0.1 * scale;
107                                      movefac = epfac * scalefac; **/
108     epsilon = epfac / scale;
109     movefac = epsilon * 0.8;
110                                       /*  error distance of eyes  */
111 err1:
112                     /* initial value of random number generator */
```

```
113       fgets(string,512,input);
114       sscanf(string,"%d",&seed);
115                                   /*  fractal h-dimension of object  */
116       fgets(string,512,input);
117       sscanf(string,"%f",&h);
118                                     /*  sigma of noise distribution  */
119       fgets(string,512,input);
120       sscanf(string,"%f",&sigma);
121       if (h>1. || h<0.) {printf("h == [0,1] reenter!! \n");
122                          goto err1;}
123     fclose(input);
124
125   /**-------------------- END OF READ PARAMETERS -------------------**/
126
127      for(i = 0;i < npolys;i++) {
128        printf(" polygon  i = %d\n",i);        /*  no. of polygons  */
129        nlines = polyarr[i][0];
130
131      initgauss(seed);
132       for(j = 1;j <= nlines;j++) {
133       printf("parameters of polyline j = %d\n",j);/*no. of polyline*/
134        vtx1 = polyarr[i][j];
135        newvtc = 0;
136        if (j == nlines) vtx2 = polyarr[i][1];
137           else  vtx2 = polyarr[i][j+1];
138
139        newvtc = mpbm(vtx1,vtx2,newvtc); /*  MidPointBMotion routine  */
140
141        if (vtx1 < vtx2) {sign = -1;ind = vtx2*(vtx2 - 1)/2 + vtx1;}
142           else {sign = 1;ind = vtx1*(vtx1 - 1)/2 + vtx2;}
143                                        /*  check ind in lookup table  */
144        if (lsrch(ltabc,ind) == T) continue;
145                                      /*  hold start vtx of vtxs sequence  */
146        addvtxs[ ltabc ].ind = ind;
147        addvtxs[ ltabc ].svtx = npts;
148        polyarr[i][0] = polyarr[i][0] + newvtc;
149
150      normpert(newvtc,sign);  /* variable normal vector perturbation  */
151
152                                     /*  hold end vtx of vtxs sequence  */
153        addvtxs[ ltabc ].evtx = npts - 1;
154        ltabc = ltabc + 1;
155        }
156      }
157
158      fraout(argv[1]);      /*  output result "epsilon" datafile *.epf  */
159
160    }
161   /**---- end of main procedure ------------------------------------**/
162
163   /**---- read point and polygon suport midpoint routine ------------**/
164
165   pline(str)
166    char  str[];
167   {
168       FILE    *input;
```

```
169       char    filename[512];
170       int     i,a,b;
171       char    string[512],str2[512];
172
173       strcpy(filename,str);
174       strcat(filename,".std");                    /*  open *.std file  */
175      input = fopen(filename,"r");
176
177                                  /*  read until EOF or "data" keyword  */
178       do {fgets(string,512,input);}
179       while ((*string != NULL)
180              && (strncmp(string,"data",4) != 0));
181       sscanf(string,"%s %d %d","data",&a,&b);
182       npts = a + 1;
183       npolys = b;
184
185       for (i = 1;i < npts;i++)       /*  read in and copy out points  */
186         { fgets(string,512,input);
187           sscanf(string,"%f %f %f",&varr[i].x,&varr[i].y,&varr[i].z);
188         }
189
190       for (i = 0;i < npolys;i++)                    /*  do polygons  */
191         { short j,k,l;
192           fgets(string,512,input);
193
194           j = 0;
195           while(string[j] == ' ') j++;             /*  ignore blanks  */
196           l = polyarr[i][0] = atoi(&string[j]);
197
198           for (k = 1; k <= polyarr[i][0]; k++)
199           { while(string[j] != ' ') j++;
200             polyarr[i][k] = atoi(&string[j]);
201             while(string[j] == ' ') j++;  /* read all vtx of polyline */
202           }
203           polyarr[i][l+1] = -1;               /* mark end of polyline */
204         }
205    fclose(input);
206  }
207  /**---------- end of read ploylines & points ----------------------**/
208
209  /**--INITGAUSS routine-set initial value of GAUSS routines----**/
210
211   initgauss(seed)
212     int seed;
213     {
214      arand = pow(2.,POWER) - 1.;
215      gfac = sqrt(12./ Nrand) / arand;
216      srand(seed);
217     }
218  /**---------- end of initgauss() ----------------------------------**/
219
220  /**--- mpbm (vtx1,vtx2,newpts,location,sigma,seed ) int newvtc;----**/
221
222  int mpbm(vtx1,vtx2,newvtc)
223   int  vtx1,vtx2;
224   int  newvtc;
```

```
225   {
226      int        i,mprecurs();
227      float      gauss(),gau;
228      float      delta[ MAXLEVEL ];
229
230      for (i = 1;i <= MAXLEVEL;i++)
231         delta[ i ] = sigma * pow(0.5,((float)i)*h)
232                       * sqrt(0.5) * sqrt(1 - pow(2.,2.*h - 2.));
233
234      newpts[0].x = varr[vtx1].x;
235      newpts[0].y = varr[vtx1].y;
236      newpts[MAXNEW-1].x = varr[vtx2].x;
237      newpts[MAXNEW-1].y = varr[vtx2].y;
238
239          newvtc = mprecurs(newpts[0],newpts[MAXNEW-1],1,delta,newvtc);
240
241      newpts[0].x = 0;
242      newpts[0].y = 0;
243      newpts[MAXNEW-1].x = 0;
244      newpts[MAXNEW-1].y = 0;
245   return(newvtc);
246   }
247  /**----- end of midpoint displacement routine ------------------**/
248
249  /**------- midpointrecursion (vtx1,vtx2.epsilon,level,delta ) ---**/
250
251  int mprecurs(svtx,evtx,level,delta,newvtc)
252    int     level,newvtc;
253    struct plocat  svtx,evtx;
254    float  delta[ MAXLEVEL ];
255   {
256     int             n,mprecurs();
257     double          dist,dx,dy;
258     float           nor,nordist(),gau;
259     struct plocat   midvtx,norm,pnorm;
260
261                                    /* distance between svtx and evtx */
262       dx = (double)(evtx.x - svtx.x);
263       dy = (double)(evtx.y - svtx.y);
264       dist = sqrt(sqr(dx) + sqr(dy));
265                                          /*  check edge finer epsilon  */
266      if (dist < epsilon) return(newvtc);
267                                                 /*  current normal  */
268       norm.x = (float)(- dy / dist);
269       norm.y = (float)(  dx / dist);
270                                               /*  normal perturbating  */
271       nor = delta[level] * nordist();
272                                     /***** moving by current normal
273                                                      nor = 0.;   *****/
274                                            /*  move midvtx location  */
275       pnorm.x = (norm.x + norm.y * nor) * movefac;
276       pnorm.y = (norm.y + norm.x * nor) * movefac;
277       gau = delta[level] * gauss();
278       midvtx.x = 0.5 * (svtx.x + evtx.x) + pnorm.x * gau;
279       midvtx.y = 0.5 * (svtx.y + evtx.y) + pnorm.y * gau;
280      if (level < MAXLEVEL )
```

```
281         {
282          —  newvtc = mprecurs(svtx,midvtx,level+1,delta,newvtc);
283             newvtc = newvtc + 1;
284             newpts[newvtc].x = midvtx.x;
285             newpts[newvtc].y = midvtx.y;
286             newvtc = mprecurs(midvtx,evtx,level+1,delta,newvtc);
287           }
288       return(newvtc);
289   }
290   /**-------- end of midpoint recurrent routine --------------------**/
291
292   /**----------- NORMAL DISTRIBUTION routine ----------------------**/
293   float nordist()
294     {
295       float  sum;
296       int    i;
297       int    nrand = 3;
298
299       sum = 0.;
300       for (i = 0;i < nrand;i++)
301           sum = sum + rand() / arand - 0.5;
302       return(sum / nrand);
303     }
304
305   /**--GAUSS routine-gauss random number generator -------------**/
306
307   float  gauss()
308      {
309       float  sum;
310       int    i;
311
312       sum = 0.;
313       for (i = 1;i <= Nrand;i++)
314           sum = sum + rand() - arand * 0.5;
315       return(gfac * sum);
316                                          /* return(fabs(gfac * sum)); */
317      }
318
319   /**-------- Linear search index of polyline ----------------------**/
320   lsrch(upper,key)
321    int   upper,key;
322    {   int   i;
323
324        i = 1;
325        while (i < upper) {
326           if (addvtxs[ i ].ind == key) return(T);
327           i = i + 1;
328        }
329        return(F);
330    }
331   /**-------- end of Linear search index of polyline ----------------**/
332
333   /**---------NORMAL PERTURBATION( newpts [ 1 to n-1 ] )---------**/
334
335   normpert(n,sign)
336     int n,sign;
```

```
337     { short i;
338                                                 /*   move by newpts   */
339       if (sign == -1)
340           for(i = n;i > 0;i--){               /* reverse new vtxs seq. */
341               varr[npts].x = newpts[i].x;
342               varr[npts].y = newpts[i].y;
343               npts = npts + 1;
344           }
345        else for(i = 1;i <= n;i++){            /* positive to vtxs seq. */
346                 varr[npts].x = newpts[i].x;
347                 varr[npts].y = newpts[i].y;
348                 npts = npts + 1;
349             }
350     }
351   /**------------- end of normperturbation ----------------------**/
352
353   /**------------- fractal datafile output routine -------------**/
354
355   fraout(str)
356     char str[];
357    {
358       FILE  *output;
359       int   i;
360       char  filename[512];
361
362       strcpy(filename,str);
363       strcat(filename,".epf");                  /*  open *.fra file  */
364       output = fopen(filename,"w");
365
366       fprintf(output,"%s %d %d \n","data",npts - 1,npolys);
367       printf("%s %d %d \n","data",npts - 1,npolys);
368       for (i = 1;i < npts; i++)    /* write all points *.x *.y *.z */
369         fprintf(output,"%f %f %f \n",varr[i].x,varr[i].y,varr[i].z);
370
371       for (i = 0; i < npolys; i++)                  /* do polygons */
372        { short j,k,vtx1,vtx2,svtx,evtx;
373          int   sign,itab,ind;
374          fprintf(output," %d",polyarr[i][0]);
375          j = 1;
376          while( polyarr[i][j] != -1) {
377            vtx1 = polyarr[i][j];
378            fprintf(output," %d",vtx1);
379            if (polyarr[i][j+1] == -1) vtx2 = polyarr[i][1];
380               else  vtx2 = polyarr[i][j+1];
381            if (vtx1 < vtx2)
382                    {sign = -1;ind = vtx2*(vtx2 - 1)/2 + vtx1;}
383               else {sign = 1;ind = vtx1*(vtx1 - 1)/2 + vtx2;}
384           itab = lfind(ltabc,ind);
385            if (itab == -1) continue;
386            svtx = addvtxs[ itab ].svtx;
387            evtx = addvtxs[ itab ].evtx;
388            if (sign == -1)
389                for(k = evtx;k >= svtx; k--) fprintf(output," %d",k);
390              else for(k=svtx;k <= evtx;k++) fprintf(output," %d",k);
391            j = j + 1;
392          }
```

```
393     errout:fprintf(output," \n");
394       }
395    fclose(output);
396    }
397   /**--------------- END OF FRAOUT ROUTINE  ----------------------**/
398
399   /**------- Linear search to find index of new points --------**/
400    lfind(upper,key)
401      int   upper,key;
402      { int   i;
403
404        i = 0;
405        while (i < upper) {
406           if (addvtxs[ i ].ind == key) return(i);
407           i = i + 1;
408        }
409        return(-1);
410      }
411   /**-------end of linear find index of new points ------------**/
```

(二)中心向量為兩端点法向量的平均向量

(/usr/local/graphics/ps300/student/skc/plpoly/mtai/mtai.c)

(1) 固定法向量

(2) 現行法向量

(3) 擾乱向量加現行法向量

為使實驗結果更真實，吾人修改 cpiem1d.C 注入合理的端点平均法向量之觀念及考慮 Animator 軟體的需求，故修改 mprecurs() 成 mprec1d()，便依隨機中点的產生位置馬上分割原 polygon 為二個或四個新的 polygons。

並且為了方便 mprec1d的處理，在 readstd() 中，加入 polynormalize()將所有 polygon 標準化。各副程式功能如下：

main()：同前。但讀取 control parameters 改由 readpar() 負責。

readstd()：同前之 plrne()，但增加了 polynormalize() 負責從輸入的平面檔案中取出含海岸

線的 polygon，並標準化後存入 celist[]（coastline edges list）送至 mpbm1d中做碎形化處理。

lfind()：從物體所有的邊中挑出全部的海岸線。

polynormalize()：從物體所有的 polygon中挑出含有海岸線的 polygon，並將之標準化。

storepoly()：存入不含海岸線之 polygon，置於 result array。

storecelist()：存含海岸線的標準 polygon，接受碎形化處理。

edgesrch()：協助 polynormalize()檢查是否為海岸線邊。

readpar()：讀入所有的 control parameters。

calvnorm()：計算所求的法向量，提供 mpbm1d 隨時計算現行法向量（current average normal vector of endpoints）

edgenormal(): 計算各 edge normals 以便利点法向量的計算。(点法向量 = 和此点有相関之所有 edge 之 normals 的平均值)。

mpbm1d(): 同前之 mpbm()。

initgauss(): 同前。

mprec1d(): 同前之 mprecurs。但現行法向量由 gauss() 控制，而擾乱量則由 uniform() 控制。

uniform(): 同前之 nordist。但 distribution type 不同。

gauss(): 同前。

fraout(): 同前。

結果存於 /usr/local/SUNISKC/mtai/taim.ras 及 ptaim.ras 同時附上，粗略外形的平面檔 *taim.std

```
/* purpose -- make a midpoint displacement method use a
     —          gauss noise function control by epsilon

main()

    void   readstd();

           int     lfind();

           void    polynormalize();

                   void   storepoly();

                   void   storecelist();

                   int    edgesrch();

    void   readpar();

    void   calvnorm();

           void edgenormal();

    void   mpbmld()

           int  initgauss();

           void mprecld();

                void    storepoly();

                void    mprecld();

                float   uniform(),gauss();

    void   fraout()

*/

#include <stdio.h>
#include <math.h>

#define sqr(x)    ((x)*(x))
#define POWER      31.
#define MAXLEVEL    5
#define MAXPTS  16384
#define MAXPOLY 16384
#define MAXVTX      4
#define Nrand       5
#define T           1
#define F           0

int       seed;           /* initial of random number seq. */
int       maxlevel;        /* user set object refine level */
float     arand,gfac;              /* constants of gauss() */
float     delta[ MAXLEVEL ];   /* variance of every level */
```

```
 57  float     sigma;                          /* initial variance */
 58  float     h;            /* indicator of fractal dimension */
 59  float     epsilon;                /* resolution of object */
 60  float     fixvl;                        /* moving factor */
 61  int       npts;                       /* number of point */
 62  int       npolys;                   /* number of polygon */
 63  int       polyc = 0;                  /* polygon counter */
 64  int       mpltc = 0;              /* result polygon counter */
 65  int       celtc = 0;          /* coastline polygon counter */
 66
 67  int    polyarr[ MAXPOLY ][ MAXVTX ];    /* polygon  array */
 68  struct  edgetype { int  ind;      /* unique index of edge */
 69                     int  cline;};  /* Is edge a coastline? */
 70  struct  { int  svtx;
 71            int  evtx;
 72            int  ovtx; } celist[ MAXPOLY ];
 73                  /* coastlines edge(svtx,evtx) list seovtx */
 74  struct  vlocat { float    x;
 75                   float    y;
 76                   float    z; } ;          /* vertice type */
 77  struct  vlocat   varr[ MAXPTS ];         /* vertice array */
 78  struct  vlocat   vnorm[ MAXPTS ]; /* normal vector of vtx */
 79
 80  main(argc,argv)
 81    int   argc;
 82    char *argv[];
 83    {int   i;
 84     void   readstd();
 85     void   readpar();
 86     void   calvnorm();
 87     void   mpbmld();
 88     void   fraout();
 89     void   storepoly();
 90
 91     readstd(argv[1]);
 92            /* read *.std and get all vertice and polylines */
 93
 94     readpar(argv[1]);
 95               /* read *.par and get all control parameters */
 96
 97     for(i = 0;i < celtc;i++)
 98        calvnorm(i);         /* computing normal of vertice */
 99
100     for (i = 0;i < MAXLEVEL;i++)
101         delta[ i ] = sigma * (float)pow(0.5,((float)i)*h)
102                      * sqrt(0.5) * sqrt(1 - pow(2.,2.*h - 2.));
103
104     for(i = 0;i < celtc;i++)
105         mpbmld(i);  /* MidPointBMotion routine for polygon */
106     npolys = polyc;
107                  /* output result "epsilon" datafile *.epf */
108     fraout(argv[1]);
109
110   }         /* end of main */
111
112  /**-- Computing Edge's normal => Vertice normal ----------**/
```

```
113  void calvnorm(ceplyno)
114    int ceplyno;
115   {
116     struct { float x,y,z; } edgenorm,pt1,pt2;
117     int isvtx,ievtx;              /* index of svtx and evtx */
118     void edgenormal();
119
120        isvtx = celist[ceplyno].svtx;
121        ievtx = celist[ceplyno].evtx;
122        pt1.x = varr[isvtx].x;
123        pt1.y = varr[isvtx].y;
124        pt1.z = varr[isvtx].z;
125        pt2.x = varr[ievtx].x;
126        pt2.y = varr[ievtx].y;
127        pt2.z = varr[ievtx].z;
128        edgenormal(&edgenorm,pt1,pt2);
129        vnorm[isvtx].x = vnorm[isvtx].x + edgenorm.x * 0.5;
130        vnorm[isvtx].y = vnorm[isvtx].y + edgenorm.y * 0.5;
131        vnorm[isvtx].z = vnorm[isvtx].z + edgenorm.z * 0.5;
132        vnorm[ievtx].x = vnorm[ievtx].x + edgenorm.x * 0.5;
133        vnorm[ievtx].y = vnorm[ievtx].y + edgenorm.y * 0.5;
134        vnorm[ievtx].z = vnorm[ievtx].z + edgenorm.z * 0.5;
135
136  }   /**  end of calvnorm() **/
137  /**---- Normal Vector of edge (vt1,vt2) ----------------**/
138  void edgenormal(norm,vt1,vt2)
139    struct { float x,y,z; } *norm,vt1,vt2;
140   {  float  dx,dy,dist;
141
142        dx = vt2.x - vt1.x;
143        dy = vt2.y - vt1.y;
144        dist = (float)sqrt(sqr(dx)+sqr(dy));
145        norm->x = -dy / dist;
146        norm->y =  dx / dist;
147        norm->z = 0.;
148   }
149
150  /**---- mpbm1d = MidPoint Brownian Motion ---------------**/
151  void mpbm1d(ceplyno)
152   int ceplyno;
153   {
154    int      iovtx,isvtx,ievtx,i;
155    void      mprec1d();
156    struct vlocat ovtx,svtx,evtx;
157
158      isvtx = celist[ ceplyno ].svtx;
159      svtx.x = varr[isvtx].x;
160      svtx.y = varr[isvtx].y;
161      svtx.z = varr[isvtx].z;
162      ievtx = celist[ ceplyno ].evtx;
163      evtx.x = varr[ievtx].x;
164      evtx.y = varr[ievtx].y;
165      evtx.z = varr[ievtx].z;
166      iovtx = celist[ ceplyno ].ovtx;
167      ovtx.x = varr[iovtx].x;
168      ovtx.y = varr[iovtx].y;
```

```
169       ovtx.z = varr[iovtx].z;
170      initgauss(seed);
171       mprecld(svtx,evtx,ovtx,isvtx,ievtx,iovtx,0);
172
173    }      /* end of mpbmld */
174
175   /**----- mprecld(svtx,evtx,ovtx,isvtx,ievtx,iovtx,0) ----**/
176   void mprecld(svtx,evtx,ovtx,isvtx,ievtx,iovtx,level)
177        int isvtx,ievtx,iovtx,level;
178        struct vlocat svtx,evtx,ovtx;
179      {
180        int     imvtx,ipvtx;     /* index of midvtx and pvtx */
181        struct vlocat mvtx,pvtx;/* locate of midvtx and pvtx */
182        struct vlocat norm;    /* normal vector of coastline */
183        struct vlocat pnorm; /* p-normal vector of coastline */
184        float   uniform(),gauss(),unif,gau;
185        float   dist,dx,dy,movefac;
186
187        mvtx.x = 0.5*(svtx.x + evtx.x);
188        mvtx.y = 0.5*(svtx.y + evtx.y);
189        mvtx.z = 0.;
190        dx = evtx.x - svtx.x;
191        dy = evtx.y - svtx.y;
192        dist = (float)sqrt(sqr(dx)+sqr(dy));
193      if(dist < epsilon) {storepoly(isvtx,ievtx,iovtx);return;}
194        norm.x = 0.5*(vnorm[isvtx].x+vnorm[ievtx].x);
195        norm.y = 0.5*(vnorm[isvtx].y+vnorm[ievtx].y);
196        norm.z = 0.;
197        unif = uniform();
198          /**unif = 0.;
199              printf("test unif = %f \n",unif); **/
200        pnorm.x = norm.x + norm.y *unif;
201        pnorm.y = norm.y - norm.x *unif;
202        pnorm.z = norm.z;
203        movefac = fixvl * dist;
204        gau = delta[level] * gauss() * movefac;
205        pvtx.x = mvtx.x + pnorm.x  * gau;
206        pvtx.y = mvtx.y + pnorm.y  * gau;
207        pvtx.z = mvtx.z ;
208      if(level>maxlevel) {storepoly(isvtx,ievtx,iovtx);return;}
209        if (gau >= 0.) {
210           varr[npts].x = mvtx.x;
211           varr[npts].y = mvtx.y;
212           varr[npts].z = mvtx.z;
213           imvtx = npts; npts = npts + 1;
214           storepoly(isvtx,imvtx,iovtx);
215           storepoly(imvtx,ievtx,iovtx);
216           vnorm[npts].x = norm.x;
217           vnorm[npts].y = norm.y;
218           vnorm[npts].z = norm.z;
219           varr[npts].x = pvtx.x;
220           varr[npts].y = pvtx.y;
221           varr[npts].z = pvtx.z;
222           ipvtx = npts; npts = npts + 1;
223           mprecld(svtx,pvtx,mvtx,isvtx,ipvtx,imvtx,level+1);
224           mprecld(pvtx,evtx,mvtx,ipvtx,ievtx,imvtx,level+1);
```

```
225       } /** end of if ( gau >= 0.)  **/
226      else {
227         vnorm[npts].x = norm.x;
228         vnorm[npts].y = norm.y;
229         vnorm[npts].z = norm.z;
230         varr[npts].x = pvtx.x;
231         varr[npts].y = pvtx.y;
232         varr[npts].z = pvtx.z;
233         ipvtx = npts; npts = npts + 1;
234         mprecld(svtx,pvtx,ovtx,isvtx,ipvtx,iovtx,level+1);
235         mprecld(pvtx,evtx,ovtx,ipvtx,ievtx,iovtx,level+1);
236       } /** end of else   **/
237    } /** end of mprecld()   **/
238
239 /**-------STORE POLYGON routine -------------------------**/
240 void storepoly(vtx1,vtx2,vtx3)
241  int  vtx1,vtx2,vtx3;
242  {
243    polyarr[ polyc ][0] = 3;
244    polyarr[ polyc ][1] = vtx1;
245    polyarr[ polyc ][2] = vtx2;
246    polyarr[ polyc ][3] = vtx3;
247    polyc = polyc + 1;
248  }          /* end of storepoly */
249
250 /**---- UINFORM DISTRIBUTION routine --------------------**/
251 float uniform()
252   {
253     float  sum;
254     int    i;
255     int    nrand = 2;
256
257     sum = 0.;
258     for (i = 0;i < nrand;i++)
259         sum = sum + rand() / arand - 0.5;
260     return(sum / nrand);
261   }        /* end of uniform */
262
263 /**------INITGAUSS routine-------------------------------**/
264  initgauss(fseed)
265    int fseed;
266    {
267     arand = pow(2.,POWER) - 1.;
268     gfac = sqrt(12.* Nrand) / arand;
269     srand(fseed);
270    }     /* end of initgauss */
271
272 /**-------GAUSS routine-----------------------------------**/
273 float  gauss()
274    {
275     float  sum;
276     int    i;
277
278     sum = 0.;
279     for (i = 0;i < Nrand;i++)
280         sum = sum + (rand() - 0.5*arand)/Nrand;
```

```
281      /** return(gfac * sum);  **/
282      return(fabs(gfac * sum));
283     }      /* end of gauss routine */
284
285  /**----fractal datafile output routine ----------**/
286  void fraout(str)
287    char str[];
288   {
289     FILE  *output;
290     int  i;
291     char  filename[512];
292
293     strcpy(filename,str);
294     strcat(filename,".epf");        /*  open *.fra file  */
295     output = fopen(filename,"w");
296
297     fprintf(output,"%s %d %d \n","data",npts - 1,npolys);
298     printf("%s %d %d \n","data",npts - 1,npolys);
299                          /* write all points *.x *.y *.z */
300     for (i = 1;i < npts; i++)
301       fprintf(output,"%f %f %f \n"
302                      ,varr[i].x,varr[i].y,varr[i].z);
303
304     for (i = 0; i < npolys; i++)          /* do polygons */
305      { int  j,k;
306        fprintf(output," %d",polyarr[i][0]);
307        for (j = 1;j <= polyarr[i][0];j++)
308            fprintf(output," %d",polyarr[i][j]);
309        fprintf(output," \n");
310      }      /* end of for loop to write polygon */
311   fclose(output);
312   }        /* end of fraout */
313
314  /**---- END OF FRAOUT ROUTINE ----------------------**/
315
316  /**---- Linear Find Coastlines ---------------------**/
317  lfind(elist,upper,key)
318    int  upper,key;
319    struct edgetype elist[ MAXPTS ];
320   { int  i=0;
321      while(i < upper) {
322           if ( elist[i].ind == key ) return(i);
323           i=i+1;
324      } /* end of while loop  */
325      return(-1);
326   }  /** end of lfind() **/
327
328  /*---- read point and polygon suport midpoint routine ----*/
329  void readstd(str)
330   char  str[];
331   {
332      FILE    *input;
333      char    filename[512];
334      int    i,a,b;
335      int    edgelc=0;                  /* edge list counter */
336      char    string[512],str2[512];
```

```
337       struct edgetype elist[ MAXPTS ];          /* edge list */
338       void    polynormalize();
339
340       strcpy(filename,str);
341       strcat(filename,".std");          /*  open *.std file  */
342      input = fopen(filename,"r");
343
344                          /* read until EOF or "data" keyword */
345       do {fgets(string,512,input);}
346       while ((*string != NULL)
347               && (strncmp(string,"data",4) != 0));
348       sscanf(string,"%s %d %d","data",&a,&b);
349       npts = a + 1;
350       npolys = b;
351
352       for (i = 1;i < npts;i++)       /* read and copy points */
353         { fgets(string,512,input);
354           sscanf(string,"%f %f %f"
355                         ,&varr[i].x,&varr[i].y,&varr[i].z);
356         }  /* end of for loop to read points */
357
358       for (i = 0;i < npolys;i++)            /*  do polygons  */
359         { int  j,k,l;
360           int  svtx,evtx,ovtx;
361           fgets(string,512,input);
362
363           j = 0;
364           while(string[j] == ' ') j++;   /*  ignore blanks  */
365           l = polyarr[i][0] = atoi(&string[j]);
366
367           for (k = 1; k <= polyarr[i][0]; k++)
368           { while(string[j] != ' ') j++;
369             polyarr[i][k] = atoi(&string[j]);
370             while(string[j] == ' ') j++;
371           }   /* end of for loop to read all vtx of polygon */
372         } /* end of for loop i--polygon */
373       fclose(input);
374
375       for (i = 0;i < npolys;i++)     /* mark coastline edges */
376         { int  k,l;
377           l = polyarr[i][0];
378           for (k=1;k<=l;k++){
379               int itab,ind,vtx1,vtx2;
380
381               vtx1 = polyarr[i][k];
382               if(k+1>l) vtx2=polyarr[i][1];
383                 else vtx2= polyarr[i][k+1];
384               if(vtx1<vtx2) ind = vtx2*(vtx2-1)/2+vtx1;
385                 else ind = vtx1*(vtx1-1)/2+vtx2;
386               itab = lfind(elist,edgelc,ind);
387               if(itab == -1) {
388                 elist[edgelc].cline = T;
389                 elist[edgelc].ind = ind;
390                 edgelc=edgelc + 1;
391                } /* end of if */
392               else elist[itab].cline = F;
```

```
393         } /* end of for-loop-k */
394       } /* ednof for loop i--poly */
395
396     polynormalize(elist,edgelc);
397
398   }     /* end of readstd */
399
400 /**- polygon normalize(elist,edgelc) => POLY(seovtx) -----*/
401 void polynormalize(elist,edgelc)
402    int  edgelc;
403    struct edgetype elist[MAXPTS];
404 {
405   int       mid;                                  /* mid vertex */
406   int       vtx1,vtx2,vtx3,i;
407   int       polytype;                  /* polygon type = 0..7 */
408   int       c1,c2,c3;
409   struct   vlocat midvtx;          /* location of mid vertex */
410
411   for(i=0;i<npolys;i++) {
412        void storecelist();
413        int edgesrch();
414
415     vtx1=polyarr[i][1];
416     vtx2=polyarr[i][2];
417     vtx3=polyarr[i][3];
418     c1=edgesrch(vtx1,vtx2,elist,edgelc);
419     c2=edgesrch(vtx2,vtx3,elist,edgelc);
420     c3=edgesrch(vtx3,vtx1,elist,edgelc);
421     polytype = c1*4 + c2*2 + c3;
422     switch ( polytype ) {
423        case 0 : /** c1=0;c2=0;c3=0;     **/
424               storepoly(vtx1,vtx2,vtx3);break;
425
426        case 1 : /** c1=0;c2=0;c3!=0;     **/
427               storecelist(vtx3,vtx1,vtx2);break;
428
429        case 2 : /** c1=0;c2!=0;c3=0;     **/
430               storecelist(vtx2,vtx3,vtx1);break;
431
432        case 3 : /** c1=0;c2!=0;c3!=0;     **/
433               midvtx.x=(varr[vtx1].x+varr[vtx2].x)/2;
434               midvtx.y=(varr[vtx1].y+varr[vtx2].y)/2;
435               midvtx.z= 0.;
436               varr[npts].x = midvtx.x;
437               varr[npts].y = midvtx.y;
438               varr[npts].z = midvtx.z;
439               mid = npts;
440               npts = npts + 1;
441               storecelist(vtx2,vtx3,mid);
442               storecelist(vtx3,vtx1,mid);break;
443
444        case 4 : /** c1!=0;c2=0;c3=0;     **/
445               storecelist(vtx1,vtx2,vtx3);break;
446
447        case 5 : /** c1!=0;c2=0;c3!=0;     **/
448               midvtx.x=(varr[vtx3].x+varr[vtx2].x)/2;
```

```
449                 midvtx.y=(varr[vtx3].y+varr[vtx2].y)/2;
450                 midvtx.z= 0.;
451                 varr[npts].x = midvtx.x;
452                 varr[npts].y = midvtx.y;
453                 varr[npts].z = midvtx.z;
454                 mid = npts;
455                 npts = npts + 1;
456                 storecelist(vtx3,vtx1,mid);
457                 storecelist(vtx1,vtx2,mid);break;
458
459           case 6 : /** c1!=0;c2!=0;c3=0;      **/
460                 midvtx.x=(varr[vtx3].x+varr[vtx1].x)/2;
461                 midvtx.y=(varr[vtx3].y+varr[vtx1].y)/2;
462                 midvtx.z= 0.;
463                 varr[npts].x = midvtx.x;
464                 varr[npts].y = midvtx.y;
465                 varr[npts].z = midvtx.z;
466                 mid = npts;
467                 npts = npts + 1;
468                 storecelist(vtx2,vtx3,mid);
469                 storecelist(vtx1,vtx2,mid);break;
470
471           case 7 : /** c1!=0;c2!=0;c3!=0;      **/
472            midvtx.x=(varr[vtx3].x+varr[vtx1].x+varr[vtx2].x)/3;
473            midvtx.y=(varr[vtx3].y+varr[vtx1].y+varr[vtx2].y)/3;
474                 midvtx.z= 0.;
475                 varr[npts].x = midvtx.x;
476                 varr[npts].y = midvtx.y;
477                 varr[npts].z = midvtx.z;
478                 mid = npts;
479                 npts = npts + 1;
480                 storecelist(vtx2,vtx3,mid);
481                 storecelist(vtx3,vtx1,mid);
482                 storecelist(vtx1,vtx2,mid);break;
483
484           default : printf("error polygon %d \n",i);
485
486           }       /* end of switch case */
487         }       /* end of for loop -polygon **/
488       }       /* end of polynormalize*/
489
490   /**---- Store Coastline Edge into list ----------------*/
491   void storecelist(svtx,evtx,ovtx)
492    int svtx,evtx,ovtx;
493    {
494      celist[ celtc ].svtx = svtx;
495      celist[ celtc ].evtx = evtx;
496      celist[ celtc ].ovtx = ovtx;
497      celtc = celtc + 1;
498    }       /* end of store coastline-edge-list */
499
500   /*----edge search routine -------------------------------*/
501   int edgesrch(vtx1,vtx2,elist,edgelc)
502      int  vtx1,vtx2,edgelc;
503      struct edgetype elist[MAXPTS];
504     { int i,ind;
```

```
505
506       if(vtx1<vtx2) ind = vtx2*(vtx2-1)/2+vtx1;
507          else ind = vtx1*(vtx1-1)/2+vtx2;
508       i=0;
509       while(i< edgelc) {
510        if(elist[i].ind == ind) return(elist[i].cline);
511        i=i+1;
512       } /* end of while loop */
513     return( -10 );
514   } /*end of edge search */
515
516  /*****---------- READ PARAMETERS ----------*****/
517   void readpar(str)
518   char  str[];
519   {
520      FILE    *input;
521      char    filename[512];
522      char    string[512];
523
524      strcpy(filename,str);
525      strcat(filename,".par");          /*  open *.par file  */
526     input = fopen(filename,"r");
527
528    do {fgets(string,512,input);}
529                          /* read until EOF or "parameters" */
530    while((*string != NULL)
531          && (strncmp(string,"parameters",10) != 0));
532
533      do {fgets(string,512,input);}
534      while ((*string != NULL)
535             && (strncmp(string,"maxlevel",8) != 0));
536      sscanf(string,"%s %d","maxlevel",&maxlevel);
537
538      do {fgets(string,512,input);}
539      while ((*string != NULL)
540             && (strncmp(string,"epsilon",7) != 0));
541      sscanf(string,"%s %f","epsilon",&epsilon);
542                                /* finer of eyes to object */
543      do {fgets(string,512,input);}
544      while ((*string != NULL)
545             && (strncmp(string,"fixvl",5) != 0));
546      sscanf(string,"%s %f","fixvl",&fixvl);
547      do {fgets(string,512,input);}
548      while ((*string != NULL)
549             && (strncmp(string,"seed",4) != 0));
550      sscanf(string,"%s %d","seed",&seed);
551                              /*initial random number generator*/
552      do {fgets(string,512,input);}
553      while ((*string != NULL)
554             && (strncmp(string,"H-Fdim",6) != 0));
555      sscanf(string,"%s %f","H-Fdim",&h);
556                              /* fractal h-dimension of object */
557      do {fgets(string,512,input);}
558      while ((*string != NULL)
559             && (strncmp(string,"sigma",5) != 0));
560      sscanf(string,"%s %f","sigma",&sigma);
```

```
561                                 /* sigma of noise distribution */
562     fclose(input);
563    }    /* end of read parameters */
564  /*345678901234567890123456789012345678901234567890123456789012345678*/
```

```
data 101 167
 1.12   1.30   0.0
 0.8    1.42   0.0
 0.6    1.6    0.0
 0.3    1.42   0.0
-0.08   1.2    0.0
-0.2    0.96   0.0
-0.48   0.6    0.0
-0.62   0.3    0.0
-0.86  -0.06   0.0
-1.0   -0.42   0.0
-1.0   -0.7    0.0
-0.92  -1.2    0.0
-0.64  -1.66   0.0
-0.4   -1.8    0.0
-0.32  -2.0    0.0
-0.28  -2.3    0.0
-0.24  -2.25   0.0
-0.11  -2.32   0.0
-0.12  -2.2    0.0
-0.1   -1.92   0.0
 0.05  -1.55   0.0
 0.4   -1.0    0.0
 0.48  -0.68   0.0
 0.58  -0.5    0.0
 0.7    0.3    0.0
 0.86   0.5    0.0
 1.0    0.8    0.0
 0.96   0.9    0.0
 0.94   1.12   0.0
 1.0    1.2    0.0
 0.4    1.3    0.0
 0.36   1.18   0.0
 0.48   1.06   0.0
 0.52   0.9    0.0
 0.6    0.94   0.0
 0.72   1.08   0.0
 0.1    1.1    0.0
 0.3    1.0    0.0
 0.4    0.8    0.0
 0.04   0.84   0.0
 0.08   0.7    0.0
 0.2    0.7    0.0
 0.3    0.62   0.0
 0.42   0.76   0.0
-0.2    0.46   0.0
 0.0    0.54   0.0
-0.48   0.26   0.0
-0.36   0.14   0.0
-0.24   0.16   0.0
 0.0    0.3    0.0
 0.4    0.42   0.0
 0.5    0.6    0.0
-0.8   -0.1    0.0
-0.65  -0.06   0.0
-0.5   -0.12   0.0
-0.4   -0.1    0.0
-0.44  -0.0    0.0
-0.9   -0.44   0.0
-0.7   -0.36   0.0
-0.5   -0.32   0.0
-0.34  -0.36   0.0
-0.26  -0.3    0.0
-0.4   -0.24   0.0
-0.9   -0.7    0.0
-0.7   -0.54   0.0
```

```
 67  -0.54 -0.62  0.0
 68  -0.5  -0.74  0.0
 69  -0.4  -0.7   0.0
 70  -0.08 -0.52  0.0
 71   0.0  -0.5   0.0
 72  -0.05 -0.44  0.0
 73  -0.16 -0.45  0.0
 74  -0.2  -0.35  0.0
 75  -0.76 -1.18  0.0
 76  -0.6  -1.0   0.0
 77  -0.58 -1.54  0.0
 78  -0.6  -1.46  0.0
 79  -0.58 -1.3   0.0
 80  -0.4  -1.18  0.0
 81  -0.2  -1.16  0.0
 82  -0.2  -1.05  0.0
 83  -0.08 -0.9   0.0
 84  -0.1  -0.8   0.0
 85   0.06 -0.66  0.0
 86  -0.2  -1.8   0.0
 87  -0.28 -1.5   0.0
 88  -0.1  -1.3   0.0
 89   0.5  -0.5   0.0
 90   0.4  -0.7   0.0
 91   0.3  -0.85  0.0
 92   0.2  -0.76  0.0
 93   0.8   0.56  0.0
 94   0.6   0.55  0.0
 95   0.3   0.2   0.0
 96   0.32  0.08  0.0
 97   0.26 -0.06  0.0
 98   0.24 -0.24  0.0
 99   0.08 -0.4   0.0
100   0.66  0.0   0.0
101   0.6  -0.2   0.0
102  -0.18 -2.28  0.0
103  3 2 1 30
104  3 2 30 36
105  3 31 4 2
106  3 36 31 2
107  3 36 32 31
108  3 36 33 32
109  3 36 35 33
110  3 35 34 33
111  3 4 3 2
112  3 39 33 34
113  3 39 38 33
114  3 38 32 33
115  3 38 37 32
116  3 37 31 32
117  3 37 4 31
118  3 37 5 4
119  3 44 43 39
120  3 43 42 39
121  3 42 38 39
122  3 42 41 38
123  3 41 40 38
124  3 40 37 38
125  3 40 6 37
126  3 6 5 37
127  3 46 42 43
128  3 46 41 42
129  3 46 45 41
130  3 45 40 41
131  3 45 6 40
132  3 45 7 6
```

```
133  3 52 51 43
134  3 51 46 43
135  3 51 50 46
136  3 50 45 46
137  3 50 49 45
138  3 49 48 45
139  3 48 47 45
140  3 47 7 45
141  3 47 8 7
142  3 47 48 57
143  3 57 56 55
144  3 55 54 57
145  3 57 54 47
146  3 54 53 47
147  3 53 8 47
148  3 53 9 8
149  3 63 62 61
150  3 63 61 60
151  3 63 55 56
152  3 63 60 55
153  3 60 59 55
154  3 59 54 55
155  3 59 53 54
156  3 59 58 53
157  3 58 10 53
158  3 53 10 9
159  3 71 70 69
160  3 72 71 69
161  3 72 69 68
162  3 73 72 68
163  3 73 61 62
164  3 73 68 61
165  3 68 67 66
166  3 68 66 61
167  3 66 60 61
168  3 66 65 60
169  3 65 59 60
170  3 65 58 59
171  3 65 64 58
172  3 64 11 58
173  3 58 11 10
174  3 75 67 68
175  3 75 65 67
176  3 65 66 67
177  3 75 74 65
178  3 74 64 65
179  3 74 12 64
180  3 64 12 11
181  3 84 83 70
182  3 83 69 70
183  3 83 68 69
184  3 83 82 81
185  3 83 81 68
186  3 81 75 68
187  3 81 80 79
188  3 81 79 75
189  3 79 78 75
190  3 78 74 75
191  3 78 77 74
192  3 77 12 74
193  3 77 76 13
194  3 77 13 12
195  3 87 86 80
196  3 86 79 80
197  3 86 78 79
198  3 86 77 78
```

```
199   3 86 76 77
200   3 86 14 76
201   3 86 85 14
202   3 85 15 14
203   3 85 20 15
204   3 20 17 15
205   3 17 16 15
206   3 19 101 17
207   3 19 18 101
208   3 20 19 17
209   3 76 14 13
210   3 21 20 85
211   3 21 85 86
212   3 21 86 87
213   3 91 21 87
214   3 80 81 82
215   3 87 80 82
216   3 91 87 82
217   3 91 82 83
218   3 91 83 84
219   3 91 90 21
220   3 90 22 21
221   3 90 89 22
222   3 89 23 22
223   3 89 88 23
224   3 88 24 23
225   3 89 90 91
226   3 98 84 70
227   3 98 91 84
228   3 98 89 91
229   3 98 88 89
230   3 98 97 88
231   3 97 24 88
232   3 97 100 24
233   3 97 96 100
234   3 96 99 100
235   3 96 95 99
236   3 95 25 99
237   3 95 94 25
238   3 94 51 25
239   3 51 92 25
240   3 93 92 51
241   3 51 52 93
242   3 92 26 25
243   3 92 27 26
244   3 52 43 44
245   3 44 39 34
246   3 52 44 34
247   3 93 52 34
248   3 93 34 35
249   3 92 93 35
250   3 27 92 35
251   3 28 27 35
252   3 28 35 36
253   3 29 28 36
254   3 30 29 36
255   3 94 50 51
256   3 94 95 50
257   3 95 96 50
258   3 96 49 50
259   3 96 97 49
260   3 97 48 49
261   3 97 57 48
262   3 97 56 57
263   3 97 63 56
264   3 97 62 63
```

265	3	97	73	62
266	3	97	98	73
267	3	98	71	73
268	3	98	70	71
269	3	71	72	73

二、二維碎形處理（/usr/local/graphics/ps300/student/skc/pmpoly）

㈠以平面法向量為中心向量，
會產生 gaps。

㈡以共用 edge normal 為中心向量
不合乎 external consistency。

㈢以端点法向量的平均向量為中心向量

由於㈠,㈡法並不合理，故吾人僅对㈢法做實驗模擬，程式為 /usr/local/graphics/ps300/student/skc/pmpoly/pm2d tai.c，此程式源於 cpnem1d.C 故亦考慮固定法向量的移動方式，僅考慮現行法向量及擾乱現行法向量等二种移动方式。其副程式功能如下：

main()：同前。

readstd()：同前，並記錄海岸線边，以便对海岸線

作水平碎形化（$\zeta=0$）。

storeaplist()：將所有 polygon 存入 active polygon list 等待處理。

lfind()：同前。

storecemd()：記錄海岸線邊之指標。

readpar()：同前。

calunorm()：同前。目的相同，但計算方式稍有不同。

X_prod()：求 polygon 之法向量。

initseed()：同前之 initgauss()。

mpbmad()：同前之 mpbm1d。但因每一三角平面都必須考慮 P_{61} 及 P_{62} 所討論之情況，故加入了程式中的 switch statement。

length()：檢查邊長是否短於 Epsilon。

emltsrh()：確定某 old edge 是否已處理過（old edge為輸入之粗略外形之邊(edge)）待處

Y — 取出處理順序過程，給 mpbm2d() 使用。

N — 由 mpbm2d() 作 New midpoint 之處理。

meltsrh()：在 mpbm2d()作分割過程所產生的新edges記錄下來，並提供和 emltsrh() 同樣的功能，但其 edge 為 new edges。

storepoly()：將處理完成的 polygon 存入 polyarr[]。有不

mprec2d1()：僅有一邊大於 Epsilon 須作分割處理(i.e.碎形化)。

mprec2d2()：有二邊大於 Epsilon。

mprec2d3()：三角形之三邊均需處理。面都，故

calmidv()：依採用之方式對每一須要處理的邊，

求出隨機中点。

uniform()：同前。

gauss()：同前。

distance()：求三点間的距離。

P_prod()：求海岸線水平碎形化的移動向量。

edgevec()：求出某边之向量。

V_unit()：將 vector 單位化。

cesrch()：確定某 edge 是否為海岸边以便作水平碎形化。

fraout()：同前

實驗結果存於 /usr/local/SUN/skc/pmpoly 下：

- ctrim*.ras 及 ctrim*.pf
- ptrim*.ras 及 ptrim*.pf 。

```
/*345678901234567890123456789012345678901234567890123456789012345678*/
/* purpose -- make a midpoint displacement method use a
              gauss noise function control by epsilon

main()

   void   readstd();

          void   storeaplist();

          int    lfind();

          void   storeceind();

   void   readpar();

   void   calvnorm()

          void   X_prod();

   void   initseed();

   void   mpbm2d()

          int    length();

          long   emltsrh(),neltsrh();

          void   storepoly();

          void   mprec2d1();mprec2d2();mprec2d3();

               long   calmidv();

                      float  uniform(),gauss();

                      float  distance();

                      void   P_prod();

                      void   edgevec();

                      void   V_unit();

                      long   cesrch();

   void   fraout()
*/

#include <stdio.h>
#include <math.h>

#define sqr(x)    ((x)*(x))
#define POWER      31.
#define MAXLEVEL   10
#define MAXNEWMP   64
```

```
 57  #define MAXEDGE 16384
 58  #define MAXPTS  16384
 59  #define MAXPOLY 16384
 60  #define MAXVTX      4
 61  #define NEWEDGE  4096
 62  #define NEWEMP     64
 63  #define Nrand       5
 64  #define NEW         1
 65  #define OLD         0
 66  #define T           1
 67  #define F           0
 68
 69
 70  int       seed;          /* initial of random number seq. */
 71  int       maxlevel;       /* user set object refine level */
 72  float     arand,gfac;             /* constants of gauss() */
 73  float     delta[ MAXLEVEL ];   /* variance of every level */
 74  float     sigma;                      /* initial variance */
 75  float     h;            /* indicator of fractal dimension */
 76  float     epsilon;                /* resolution of object */
 77  float     fixvl;                        /* moving factor */
 78  long     npts;                       /* number of point */
 79  long     npolys;                    /* number of polygon */
 80  long     polyc = 0;             /* result polygon counter */
 81  long     apltc = 0;             /* active polygon counter */
 82  long     celtc = 0;             /* coastline edge counter */
 83
 84  long     emltc = 0;          /* old edge counter of object */
 85  long     neltc = 0;          /* new edge counter of object */
 86
 87                              /** new midpts of new edge list **/
 88  long  nelist[ NEWEDGE ][ NEWEMP ];
 89  long  neltind[ NEWEDGE ];          /** index of edge list **/
 90                              /** new midpts of old edge list **/
 91  long  emlist[ MAXEDGE ][ MAXNEWMP ];
 92  long  emltind[ MAXEDGE ];          /** index of edge list **/
 93  long  polyarr[ MAXPOLY ][ MAXVTX ];   /** polygon  array **/
 94  int   plytype[ MAXPOLY ];              /** polygon  type **/
 95
 96  struct { long  vtx1;
 97           long  vtx2;
 98           long  vtx3; } aplist[ MAXPOLY ];
 99                            /**  active polygon list 123vtx  **/
100  struct vlocat { float    x;
101                  float    y;
102                  float    z; } ;       /**  vertice type  **/
103  struct vlocat varr[ MAXPTS ];         /** vertice array  **/
104  struct vlocat vnorm[ MAXPTS ];/**  normal vector of vtx  **/
105
106  long    ceindlt[ MAXEDGE ];
107                         /**  index of coastline-edge list  **/
108
109  main(argc,argv)
110    int   argc;
111    char *argv[];
112    {long   i;
```

```
113     void   readstd();
114     void   readpar();
115     void   calvnorm();
116     void   mpbm2d();
117     void   fraout();
118     void   storepoly();
119
120            /* read *.std and get all vertice and polylines */
121     readstd(argv[1]);
122
123          /* read *.par and get all perturbation parameters */
124     readpar(argv[1]);
125
126     calvnorm();           /** computing normal of vertice **/
127
128                             /**  Variance of every level  **/
129     for (i = 0;i < MAXLEVEL;i++)
130         delta[ i ] = sigma * (float)pow(0.5,((float)i)*h)
131                    * sqrt(0.5) * sqrt(1 - pow(2.,2.*h - 2.));
132
133   initseed(seed);
134     for(i = 0;i < apltc;i++)
135         mpbm2d(i);  /* MidPointBMotion routine for polygon */
136     npolys = polyc;
137
138     fraout(argv[1]);      /* output epsilon-datafile *.epf */
139
140   }         /* end of main */
141
142  /*- read point and polygon suport midpoint routine -------*/
143  void readstd(str)
144   char  str[];
145   {
146      FILE    *input;
147      char    filename[512];
148      char    string[512],str2[512];
149      void    storeaplist();
150      long    i,a,b;
151      long    edgelc=0;                  /* edge list counter */
152      struct  { int cline;            /* Is edge a coastline ? */
153                long ind;              /* unique index of edge */
154              } elist[ MAXEDGE ];     /* edge list of object */
155      void    storecdind();
156
157
158      strcpy(filename,str);
159      strcat(filename,".std");          /*  open *.std file  */
160     input = fopen(filename,"r");
161                          /* read until EOF or "data" keyword */
162      do {fgets(string,512,input);}
163      while ((*string != NULL)
164              && (strncmp(string,"data",4) != 0));
165      sscanf(string,"%s %d %d","data",&a,&b);
166      npts = a + 1;
167      npolys = b;
168                            /* read in and copy out points */
```

```
169       for (i = 1;i < npts;i++)
170         { fgets(string,512,input);
171           sscanf(string,"%f %f %f"
172                         ,&varr[i].x,&varr[i].y,&varr[i].z);
173         } /* end of for loop to read points */
174
175       for (i = 0;i < npolys;i++)            /*  do polygons  */
176         { long  j,k,l;
177           long  vtx1,vtx2,vtx3;
178           fgets(string,512,input);
179
180           j = 0;
181           while(string[j] == ' ') j++;      /* ignore blanks */
182           l = polyarr[i][0] = atoi(&string[j]);
183
184           for (k = 1; k <= polyarr[i][0]; k++)
185           { while(string[j] != ' ') j++;
186             polyarr[i][k] = atoi(&string[j]);
187             while(string[j] == ' ') j++;
188           }   /* end of for loop to read all vtx of polygon */
189         vtx1 = polyarr[i][1];
190         vtx2 = polyarr[i][2];
191         vtx3 = polyarr[i][3];
192         storeaplist(vtx1,vtx2,vtx3);
193         } /* end of for loop i--polygon */
194     fclose(input);
195
196       for (i = 0;i < npolys;i++)            /*  do polygons  */
197         { int  k,l;
198           l = polyarr[i][0];
199           for (k=1;k<=l;k++){
200               int itab,ind,vtx1,vtx2;
201
202               vtx1 = polyarr[i][k];
203               if(k+1>l) vtx2=polyarr[i][1];
204                 else vtx2= polyarr[i][k+1];
205               if(vtx1<vtx2) ind = vtx2*(vtx2-1)/2+vtx1;
206                 else ind = vtx1*(vtx1-1)/2+vtx2;
207               itab = lfind(elist,edgelc,ind);
208               if(itab == -1) {
209                 elist[edgelc].cline = T;
210                 elist[edgelc].ind = ind;
211                 edgelc=edgelc + 1;
212                } /* end of if */
213               else elist[itab].cline = F;
214           } /* end of for-loop-k */
215         } /* ednof for loop i--poly */
216       storeceind(elist,edgelc);
217     }     /* end of readstd */
218
219   /**- STORE Active Polygon List routine ------------------**/
220   void storeaplist(vtx1,vtx2,vtx3)
221    long vtx1,vtx2,vtx3;
222    {
223      aplist[ apltc ].vtx1 = vtx1;
224      aplist[ apltc ].vtx2 = vtx2;
```

```
225     aplist[ apltc ].vtx3 = vtx3;
226     apltc = apltc + 1;
227   }       /* end of store mountain polygon list */
228
229  /*--------- linear searching ----------------------------------------*/
230  lfind(elist,upper,key)
231    long upper,key;
232     struct  { int cline;
233               long ind; } elist[ MAXEDGE ];
234   { int  i=0;
235      while(i < upper) {
236           if ( elist[i].ind == key ) return(i);
237           i=i+1;
238      } /* end of while loop  */
239      return(-1);
240   }  /** end of lfind() **/
241
242  /**- store coastline edge -------------------------------**/
243  long storeceind(elist,edgelc)
244     long    edgelc;
245      struct  { int cline;
246                long ind; } elist[ MAXEDGE ];
247    { int i=0;
248
249      while(i< edgelc) {
250       if(elist[i].cline == T) {ceindlt[celtc] = elist[i].ind;
251            printf("test elist[%d].ind = %d /n"
252                                          ,i,elist[i].ind);}
253       celtc = celtc + 1;
254       i=i+1;
255      } /* end of while loop */
256   } /*end of edge search */
257
258  /**-------- READ PARAMETERS -----------------------------**/
259   void readpar(str)
260   char  str[];
261   {
262      FILE    *input;
263      char    filename[512];
264      char    string[512];
265
266      strcpy(filename,str);
267      strcat(filename,".par");          /*  open *.par file  */
268     input = fopen(filename,"r");
269                              /*read until EOF or "parameters"*/
270    do {fgets(string,512,input);}
271    while((*string != NULL)
272           && (strncmp(string,"parameters",10) != 0));
273
274      do {fgets(string,512,input);}
275      while ((*string != NULL)
276              && (strncmp(string,"maxlevel",8) != 0));
277      sscanf(string,"%s %d","maxlevel",&maxlevel);
278
279      do {fgets(string,512,input);}
280      while ((*string != NULL)
```

```
281             && (strncmp(string,"epsilon",7) != 0));
282     sscanf(string,"%s %f","epsilon",&epsilon);
283                             /* distance of eyes to object */
284     do {fgets(string,512,input);}
285     while ((*string != NULL)
286             && (strncmp(string,"fixvl",5) != 0));
287     sscanf(string,"%s %f","fixvl",&fixvl);
288
289     do {fgets(string,512,input);}
290     while ((*string != NULL)
291             && (strncmp(string,"seed",4) != 0));
292     sscanf(string,"%s %d","seed",&seed);
293                 /* initial value random number generator */
294     do {fgets(string,512,input);}
295     while ((*string != NULL)
296             && (strncmp(string,"H-Fdim",6) != 0));
297     sscanf(string,"%s %f","H-Fdim",&h);
298                           /* fractal h-dimension of object */
299     do {fgets(string,512,input);}
300     while ((*string != NULL)
301             && (strncmp(string,"sigma",5) != 0));
302     sscanf(string,"%s %f","sigma",&sigma);
303                             /* sigma of noise distribution */
304     if (h>1. || h<0.) printf("h == [0,1] reenter!! \n");
305   fclose(input);
306  }    /* end of read parameters */
307
308 /**---- Computing Polygon normal => Vertice normal -------*/
309 void calvnorm()
310  {
311    struct vlocat polynorm[MAXPOLY];
312    struct { float x,y,z; } norm,pt1,pt2,pt3;
313    int    i;
314
315    for(i = 0;i < npolys;i++) { int vt1,vt2,vt3;
316
317       vt1 = polyarr[i][1];
318       vt2 = polyarr[i][2];
319       vt3 = polyarr[i][3];
320       pt1.x = varr[vt1].x;
321       pt1.y = varr[vt1].y;
322       pt1.z = varr[vt1].z;
323       pt2.x = varr[vt2].x;
324       pt2.y = varr[vt2].y;
325       pt2.z = varr[vt2].z;
326       pt3.x = varr[vt3].x;
327       pt3.y = varr[vt3].y;
328       pt3.z = varr[vt3].z;
329      X_prod(&polynorm[i],pt1,pt2,pt3) ;
330       vnorm[vt1].x = vnorm[vt1].x + polynorm[i].x;
331       vnorm[vt1].y = vnorm[vt1].y + polynorm[i].y;
332       vnorm[vt1].z = vnorm[vt1].z + polynorm[i].z;
333       vnorm[vt2].x = vnorm[vt2].x + polynorm[i].x;
334       vnorm[vt2].y = vnorm[vt2].y + polynorm[i].y;
335       vnorm[vt2].z = vnorm[vt2].z + polynorm[i].z;
336       vnorm[vt3].x = vnorm[vt3].x + polynorm[i].x;
```

```
337         vnorm[vt3].y = vnorm[vt3].y + polynorm[i].y;
338         vnorm[vt3].z = vnorm[vt3].z + polynorm[i].z;
339       }
340       for(i = 1;i < npts;i++) { float ndist;
341          ndist =(float)sqrt(sqr(vnorm[i].x)
342                     + sqr(vnorm[i].y) + sqr(vnorm[i].z));
343          vnorm[i].x = vnorm[i].x / ndist;
344          vnorm[i].y = vnorm[i].y / ndist;
345          vnorm[i].z = vnorm[i].z / ndist;
346       }
347  }/* end of calvnorm() */
348
349  /* ---------------------- X_PROD ------------------------- */
350  X_prod(norm,pt1,pt2,pt3)          /* vector cross-product */
351       struct { float x,y,z; } *norm,pt1,pt2,pt3;
352  {    pt1.x = pt2.x - pt1.x;
353       pt1.y = pt2.y - pt1.y;
354       pt1.z = pt2.z - pt1.z;
355       pt2.x = pt3.x - pt2.x;
356       pt2.y = pt3.y - pt2.y;
357       pt2.z = pt3.z - pt2.z;
358       norm->x = pt1.y*pt2.z - pt1.z*pt2.y;
359       norm->y = pt1.z*pt2.x - pt1.x*pt2.z;
360       norm->z = pt1.x*pt2.y - pt1.y*pt2.x;
361  }
362
363  /**---------------- OLD_EDGE_STORE_LIST ------------------**/
364  long emltsrh(vtx1,vtx2)
365     long   vtx1,vtx2;
366     { long  ind,i;
367
368       if (vtx1 < vtx2) ind = vtx2*(vtx2 - 1)/2 + vtx1;
369           else ind = vtx1*(vtx1 - 1)/2 + vtx2;
370       i = 0;
371       while (i < emltc) {
372          if (emltind[ i ] == ind && emlist[i][0] == 0)
373             {emlist[i][0]=1;return(i);}
374          i = i + 1;
375       }     /* end of while loop */
376       emltind[emltc] = ind;
377       emlist[emltc][0] = 0;
378       emltc = emltc + 1;
379       return(emltc-1);
380     }
381
382  /**---------------- NEW_EDGE_STORE_LIST ------------------**/
383  long neltsrh(vtx1,vtx2)
384     long   vtx1,vtx2;
385     { long  ind,i=0;
386
387       if (vtx1 < vtx2) ind = vtx2*(vtx2 - 1)/2 + vtx1;
388           else ind = vtx1*(vtx1 - 1)/2 + vtx2;
389       while (i < neltc) {
390          if (neltind[ i ] == ind) {nelist[i][0]=1;return(i);}
391          i = i + 1;
392       }     /* end of while loop */
```

```
393       neltind[neltc] = ind;
394       nelist[neltc][0] = 0;
395       neltc = neltc + 1;
396       return(neltc-1);
397     }
398
399  /**----- mpbm2d = MidPoint Brownian Motion ---------------**/
400  void mpbm2d(apolyno)
401   long apolyno;
402   {
403    long      ivtx1,ivtx2,ivtx3;
404    long      ind1,ind2,ind3;   /* index of polygon's edges */
405    long      i,j;
406                          /* three type recurrsive procedure */
407    void      mprec2d1(),mprec2d2(),mprec2d3();
408
409    int       length(),l12,l23,l31;        /* type of edges */
410    int       polytype;             /* polygon division type */
411    int       ptype;                          /* polygon type */
412    long      calmidv();
413    long      emltsrh(),neltsrh();
414
415      neltc = 0;
416      for (i = 0;i < NEWEDGE;i++) neltind[ i ] = 0;
417      for (i = 0;i < NEWEDGE;i++)
418          for (j = 0;j < NEWEMP;j++) nelist[ i ][ j ] = 0;
419      ptype = 0;
420      ivtx1 = aplist[ apolyno ].vtx1;
421      ivtx2 = aplist[ apolyno ].vtx2;
422      ivtx3 = aplist[ apolyno ].vtx3;
423                          /** check L(i,j)=0 smaller epsilon **/
424      l12 = length(ivtx1,ivtx2);
425      l23 = length(ivtx2,ivtx3);
426      l31 = length(ivtx3,ivtx1);
427                                  /* search emlist ind-Eij */
428    if (l12 != 0) ind1 = emltsrh(ivtx1,ivtx2);
429    if (l23 != 0) ind2 = emltsrh(ivtx2,ivtx3);
430    if (l31 != 0) ind3 = emltsrh(ivtx3,ivtx1);
431
432      polytype = l12*4 + l23*2 + l31;
433      switch ( polytype ) {
434         case 0 : /** l12=0;l23=0;l31=0;     **/
435                storepoly(ptype,ivtx1,ivtx2,ivtx3);break;
436
437         case 1 : /** l12=0;l23=0;l31!=0     **/
438                mprec2d1(ivtx3,ivtx1,ivtx2,0
439                              ,ind3,1,OLD);break;
440
441         case 2 : /** l12=0;l23!=0;l31=0     **/
442                mprec2d1(ivtx2,ivtx3,ivtx1,0
443                              ,ind2,1,OLD);break;
444
445         case 3 : /** l12=0;l23!=0;l31!=0    **/
446                mprec2d2(ivtx1,ivtx2,ivtx3,0
447                              ,ind2,1,OLD,ind3,1,OLD);break;
448
```

```
449         case 4 : /** l12!=0;l23=0;l31=0      **/
450                  mprec2d1(ivtx1,ivtx2,ivtx3,0
451                                ,ind1,1,OLD);break;
452
453         case 5 : /** l12!=0;l23=0;l31!=0      **/
454                  mprec2d2(ivtx2,ivtx3,ivtx1,0
455                                ,ind3,1,OLD,ind1,1,OLD);break;
456
457         case 6 : /** l12!=0;l23!=0;l31=0      **/
458                  mprec2d2(ivtx3,ivtx1,ivtx2,0
459                                ,ind1,1,OLD,ind2,1,OLD);break;
460
461         case 7 : /** l12!=0;l23!=0;l31!=0     **/
462                  mprec2d3(ivtx1,ivtx2,ivtx3,0
463                          ,ind1,1,OLD,ind2,1,OLD,ind3,1,OLD);
464                  break;
465       }       /* end of switch case */
466
467   }       /* end of mpbm2d */
468
469  /*-------------------- MP_REC_2D_1 ------------------------*/
470  void mprec2d1(isvtx,ievtx,iovtx,level,ind,tptr,chon)
471    int     level;
472    int     tptr;      /* pointer of random midpoint in edges */
473    int     chon;                    /* check old or new edge */
474    long    ind,isvtx,ievtx,iovtx;
475    {
476      int       lsm,lem,lom;                  /* type of edge */
477      int       tptrl,tptrr;/*left & right midpoint pointer */
478      int       polytype1;        /* polygon 1 division type */
479      int       polytype2;        /* polygon 2 division type */
480      int       ptype;                        /* polygon type */
481      long      imvtx;
482      long      nind;                        /* index of edge */
483
484      ptype = 1;
485                           /**  printf(" ptype = %d \n",ptype);
486                              printf(" lsvel = %d \n",level);**/
487     if (level > maxlevel)
488        { storepoly(ptype,isvtx,ievtx,iovtx);return;}
489      imvtx = calmidv(isvtx,ievtx,level,ind,tptr,chon);
490                           /** check L(i,j)=0 smaller epsilon **/
491      lsm = length(isvtx,imvtx);
492      lom = length(iovtx,imvtx);
493      lem = length(ievtx,imvtx);
494      if (lom != 0) nind = neltsrh(iovtx,imvtx);
495      if ( chon == NEW )
496         { if (nelist[ind][0] == 0)
497              {tptrl = tptr*2;tptrr = tptr*2 + 1;}
498            else  {tptrl = tptr*2 + 1;tptrr = tptr*2;}
499         }
500      else {if (emlist[ind][0] == 0)
501                {tptrl = tptr*2;tptrr = tptr*2 + 1;}
502              else  {tptrl = tptr*2 + 1;tptrr = tptr*2;}
503           }
504      polytype1 = lsm*2 + lom;
```

```
505       switch ( polytype1 ) {
506        case 0 : /** lsm=0;lom=0;      **/
507                 storepoly(ptype,isvtx,imvtx,iovtx);break;
508
509          case 1 : /** lsm=0;lom!=0      **/
510                 mprec2d1(imvtx,iovtx,isvtx,level+1
511                                 ,nind,1,NEW);break;
512
513          case 2 : /** lsm!=0;lom=0      **/
514                 mprec2d1(isvtx,imvtx,iovtx,level+1
515                                 ,ind,tptr1,chon);break;
516
517          case 3 : /** lsm!=0;lom!=0     **/
518                 mprec2d2(iovtx,isvtx,imvtx,level+1
519                         ,ind,tptr1,chon,nind,1,NEW);break;
520
521       }       /* end of switch case */
522
523       polytype2 = lom*2 + lem;
524       switch ( polytype2 ) {
525          case 0 : /** lom=0;lem=0;      **/
526                 storepoly(ptype,iovtx,imvtx,ievtx);break;
527
528          case 1 : /** lom=0;lem!=0      **/
529                 mprec2d1(imvtx,ievtx,iovtx,level+1
530                                 ,ind,tptrr,chon);break;
531
532          case 2 : /** lom!=0;lem=0      **/
533                 mprec2d1(iovtx,imvtx,ievtx,level+1
534                                 ,nind,1,NEW);break;
535
536          case 3 : /** lom!=0;lem!=0     **/
537                 mprec2d2(ievtx,iovtx,imvtx,level+1
538                         ,nind,1,NEW,ind,tptrr,chon);break;
539
540       }       /* end of switch case */
541      if (lom != 0) nelist[nind][0] = 1;
542    }       /* end of mpbm2d1 */
543
544  /*-------------------- MP_REC_2D_2 -------------------------*/
545  void mprec2d2(ievtx,isvtx,iovtx,level
546                      ,inds,tpts,chons,inde,tpte,chone)
547    int     level;
548    int     tpts,tpte;              /* pointer of svtx & evtx */
549    int     chons,chone;                 /* check old or new */
550    long    inds,inde,isvtx,ievtx,iovtx;
551    {
552                          /* left or right pointer of midpoint */
553       int        tptsl,tptsr,tptel,tpter;
554       int        lssm,losm,leem,loem,lesm,lsme;/*type of edge*/
555       int        ptype;                          /* polygon type */
556                             /* division type of polygon 1,2,3 */
557       int        polytype1,polytype2,polytype3;
558       long       ismvtx,iemvtx;
559       long       n1ind,n2ind;          /* index of new edges */
560
```

```
561       ptype = 2;
562                          /** printf(" ptype = %d \n",ptype);
563                           printf(" lsvel = %d \n",level); **/
564      if (level > maxlevel )
565                    { storepoly(ptype,ievtx,isvtx,iovtx);return;}
566       ismvtx = calmidv(isvtx,iovtx,level,inds,tpts,chons);
567       iemvtx = calmidv(ievtx,iovtx,level,inde,tpte,chone);
568                         /** check L(i,j)=0 smaller epsilon **/
569       lssm = length(isvtx,ismvtx);
570       losm = length(iovtx,ismvtx);
571       leem = length(ievtx,iemvtx);
572       loem = length(iovtx,iemvtx);
573       lesm = length(ievtx,ismvtx);
574       lsme = length(ismvtx,iemvtx);
575       if (lesm != 0) n1ind = neltsrh(ievtx,ismvtx);
576       if (lsme != 0) n2ind = neltsrh(ismvtx,iemvtx);
577       if ( chons == NEW )
578          { if (nelist[inds][0] == 0)
579               {tptsl = tpts*2;tptsr = tpts*2 + 1;}
580             else  {tptsl = tpts*2 + 1;tptsr = tpts*2;}
581          }
582       else {if (emlist[inds][0] == 0)
583                {tptsl = tpts*2;tptsr = tpts*2 + 1;}
584              else  {tptsl = tpts*2 + 1;tptsr = tpts*2;}
585            }
586       if ( chone == NEW )
587          { if (nelist[inde][0] == 0)
588               {tptel = tpte*2;tpter = tpte*2 + 1;}
589             else  {tptel = tpte*2 + 1;tpter = tpte*2;}
590          }
591       else {if (emlist[inde][0] == 0)
592                {tptel = tpte*2;tpter = tpte*2 + 1;}
593              else  {tptel = tpte*2 + 1;tpter = tpte*2;}
594            }
595       polytype1 = lssm*2 + lesm;
596       switch ( polytype1 ) {
597          case 0 : /** lssm=0;lesm=0;      **/
598                   storepoly(ptype,isvtx,ismvtx,ievtx);break;
599
600          case 1 : /** lssm=0;lesm!=0      **/
601                   mprec2d1(ismvtx,ievtx,isvtx,level+1
602                                ,n1ind,1,NEW);break;
603
604          case 2 : /** lssm!=0;lesm=0      **/
605                   mprec2d1(isvtx,ismvtx,ievtx,level+1
606                               ,inds,tptsl,chons);break;
607
608          case 3 : /** lssm!=0;lesm!=0     **/
609                   mprec2d2(ievtx,isvtx,ismvtx,level+1
610                           ,inds,tptsl,chons,n1ind,1,NEW);break;
611
612       }      /* end of switch case */
613
614       polytype2 = lesm*4 + lsme*2 + leem;
615       switch ( polytype2 ) {
616          case 0 : /** lesm=0;lsme=0;leem=0;      **/
```

```
617                storepoly(ptype,ievtx,ismvtx,iemvtx);break;
618
619             case 1 : /** lesm=0;lsme=0;leem!=0;      **/
620                mprec2d1(iemvtx,ievtx,ismvtx,level+1
621                               ,inde,tpter,chone);break;
622
623             case 2 : /** lesm=0;lsme!=0;leem=0;      **/
624                mprec2d1(ismvtx,iemvtx,ievtx,level+1
625                               ,n2ind,1,NEW);break;
626
627             case 3 : /** lesm=0;lsme!=0;leem!=0;      **/
628                mprec2d2(ievtx,ismvtx,iemvtx,level+1
629                        ,n2ind,1,NEW,inde,tpter,chone);break;
630
631             case 4 : /** lesm!=0;lsme=0;leem=0;      **/
632                mprec2d1(ievtx,ismvtx,iemvtx,level+1
633                               ,n1ind,1,NEW);break;
634
635             case 5 : /** lesm!=0;lsme=0;leem!=0;      **/
636                mprec2d2(ismvtx,iemvtx,ievtx,level+1
637                        ,inde,tpter,chone,n1ind,1,NEW);break;
638
639             case 6 : /** lesm!=0;lsme!=0;leem=0;      **/
640                mprec2d2(iemvtx,ievtx,ismvtx,level+1
641                               ,n1ind,1,NEW,n2ind,1,NEW);break;
642
643             case 7 : /** lesm!=0;lsme!=0;leem!=0;      **/
644                mprec2d3(ievtx,ismvtx,iemvtx,level+1
645                   ,n1ind,1,NEW,n2ind,1,NEW,inde,tpter,chone);
646                break;
647
648          }      /* end of switch case */
649
650          polytype3 = loem*4 + lsme*2 + losm;
651          switch ( polytype3 ) {
652             case 0 : /** loem=0;lsme=0;losm=0;      **/
653                storepoly(ptype,iovtx,iemvtx,ismvtx);break;
654
655             case 1 : /** loem=0;lsme=0;losm!=0;      **/
656                mprec2d1(ismvtx,iovtx,iemvtx,level+1
657                               ,inds,tptsr,chons);break;
658
659             case 2 : /** loem=0;lsme!=0;losm=0;      **/
660                mprec2d1(iemvtx,ismvtx,iovtx,level+1
661                               ,n2ind,1,NEW);break;
662
663             case 3 : /** loem=0;lsme!=0;losm!=0;      **/
664                mprec2d2(iovtx,iemvtx,ismvtx,level+1
665                        ,n2ind,1,NEW,inds,tptsr,chons);break;
666
667             case 4 : /** loem!=0;lsme=0;losm=0;      **/
668                mprec2d1(iovtx,iemvtx,ismvtx,level+1
669                               ,inde,tptel,chone);break;
670
671             case 5 : /** loem!=0;lsme=0;losm!=0;      **/
672                mprec2d2(iemvtx,ismvtx,iovtx,level+1
```

```
673                         ,inds,tptsr,chons,inde,tptel,chone);
674             break;
675
676           case 6 : /** loem!=0;lsme!=0;losm=0;      **/
677                 mprec2d2(ismvtx,iovtx,iemvtx,level+1
678                         ,inde,tptel,chone,n2ind,1,NEW);break;
679
680           case 7 : /** loem!=0;lsme!=0;losm!=0;      **/
681                mprec2d3(iovtx,iemvtx,ismvtx,level+1
682                        ,inde,tptel,chone,n2ind,1,NEW,inds,tptsr
683                        ,chons);break;
684
685       }       /* end of switch case */
686            /*printf("test mprec2d 2--level = %d\n",level); **/
687      if (lesm != 0) nelist[n1ind][0] = 1 ;
688      if (lsme != 0) nelist[n2ind][0] = 1 ;
689    }       /* end of mpbm2d */
690
691  /*--------- MP_REC_2D_3 ----------------------------------*/
692  void mprec2d3(ivtx1,ivtx2,ivtx3,level
693                ,ind1,tp1,chon1,ind2,tp2,chon2,ind3,tp3,chon3)
694     int     level;
695     int     chon1,chon2,chon3;            /* check old or new */
696     int     tp1,tp2,tp3;          /* pointer of three vtx1,2,3 */
697     long    ind1,ind2,ind3,ivtx1,ivtx2,ivtx3;
698     {
699       long       imvtx1,imvtx2,imvtx3;
700       long       n1ind,n2ind,n3ind; /* index of new edges */
701              /* left or right pointer of recursive mid point */
702       int        tp11,tp21,tp31,tp1r,tp2r,tp3r;
703                                                  /* check edge */
704       int        l1m1,l2m1,l2m2,l3m2,l3m3,l1m3,lm31,lm12,lm23;
705           /* polygon type & division type of polygon1,2,3,4 */
706       int        ptype,polytype1,polytype2,polytype3,polytype4;
707
708       ptype = 3;
709                           /**  printf(" ptype = %d \n",ptype);
710                             printf(" lsvel = %d \n",level); **/
711      if (level > maxlevel)
712         {storepoly(ptype,ivtx1,ivtx2,ivtx3);return;}
713       imvtx1 = calmidv(ivtx1,ivtx2,level,ind1,tp1,chon1);
714       imvtx2 = calmidv(ivtx2,ivtx3,level,ind2,tp2,chon2);
715       imvtx3 = calmidv(ivtx3,ivtx1,level,ind3,tp3,chon3);
716                            /* check L(i,j)=0 smaller epsilon */
717       l1m1 = length(ivtx1,imvtx1);
718       l2m1 = length(ivtx2,imvtx1);
719       l2m2 = length(ivtx2,imvtx2);
720       l3m2 = length(ivtx3,imvtx2);
721       l3m3 = length(ivtx3,imvtx3);
722       l1m3 = length(ivtx1,imvtx3);
723       lm31 = length(imvtx3,imvtx1);
724       lm12 = length(imvtx1,imvtx2);
725       lm23 = length(imvtx2,imvtx3);
726       if (lm31 != 0) n1ind = neltsrh(imvtx3,imvtx1);
727       if (lm12 != 0) n2ind = neltsrh(imvtx1,imvtx2);
728       if (lm23 != 0) n3ind = neltsrh(imvtx2,imvtx3);
```

```
729       if ( chon1 == NEW )
730         { if (nelist[ind1][0] == 0)
731             {tp1l = tp1*2;tp1r = tp1*2 + 1;}
732           else  {tp1l = tp1*2 + 1;tp1r = tp1*2;}
733         }
734       else {if (emlist[ind1][0] == 0)
735               {tp1l = tp1*2;tp1r = tp1*2 + 1;}
736             else  {tp1l = tp1*2 + 1;tp1r = tp1*2;}
737           }
738       if ( chon2 == NEW )
739          { if (nelist[ind2][0] == 0)
740              {tp2l = tp2*2;tp2r = tp2*2 + 1;}
741            else  {tp2l = tp2*2 + 1;tp2r = tp2*2;}
742          }
743       else {if (emlist[ind2][0] == 0)
744               {tp2l = tp2*2;tp2r = tp2*2 + 1;}
745             else  {tp2l = tp2*2 + 1;tp2r = tp2*2;}
746           }
747       if ( chon3 == NEW )
748         {  if (nelist[ind3][0] == 0)
749              {tp3l = tp3*2;tp3r = tp3*2 + 1;}
750            else  {tp3l = tp3*2 + 1;tp3r = tp3*2;}
751         }
752       else {if (emlist[ind3][0] == 0)
753               {tp3l = tp3*2;tp3r = tp3*2 + 1;}
754             else  {tp3l = tp3*2 + 1;tp3r = tp3*2;}
755           }
756       polytype1 = l1m1*4 + lm31*2 + l1m3;
757       switch ( polytype1 ) {
758          case 0 : /** l1m1=0;lm31=0;l1m3=0;      **/
759                   storepoly(ptype,ivtx1,imvtx1,imvtx3);break;
760
761          case 1 : /** l1m1=0;lm31=0;l1m3!=0;     **/
762                   mprec2d1(imvtx3,ivtx1,imvtx1,level+1
763                                  ,ind3,tp3r,chon3);break;
764
765          case 2 : /** l1m1=0;lm31!=0;l1m3=0;     **/
766                   mprec2d1(imvtx1,imvtx3,ivtx1,level+1
767                                  ,n1ind,1,NEW);break;
768
769          case 3 : /** l1m1=0;lm31!=0;l1m3!=0;     **/
770                   mprec2d2(ivtx1,imvtx1,imvtx3,level+1
771                          ,n1ind,1,NEW,ind3,tp3r,chon3);break;
772
773          case 4 : /** l1m1!=0;lm31=0;l1m3=0;     **/
774                   mprec2d1(ivtx1,imvtx1,imvtx3,level+1
775                                  ,ind1,tp1l,chon1);break;
776
777          case 5 : /** l1m1!=0;lm31=0;l1m3!=0;     **/
778                   mprec2d2(imvtx1,imvtx3,ivtx1,level+1
779                         ,ind3,tp3r,chon3,ind1,tp1l,chon1);break;
780
781          case 6 : /** l1m1!=0;lm31!=0;l1m3=0;     **/
782                   mprec2d2(imvtx3,ivtx1,imvtx1,level+1
783                          ,ind1,tp1l,chon1,n1ind,1,NEW);break;
784
```

```
785         case 7 : /** l1m1!=0;lm31!=0;l1m3!=0;      **/
786                  mprec2d3(ivtx1,imvtx1,imvtx3,level+1
787                  ,ind1,tp1l,chon1,n1ind,1,NEW,ind3,tp3r,chon3);
788                  break;
789       }      /* end of switch case */
790
791       polytype2 = l2m2*4 + lm12*2 + l2m1;
792       switch ( polytype2 ) {
793          case 0 : /** l2m2=0;lm12=0;l2m1=0;      **/
794                   storepoly(ptype,ivtx2,imvtx2,imvtx1);break;
795
796          case 1 : /** l2m2=0;lm12=0;l2m1!=0;      **/
797                   mprec2d1(imvtx1,ivtx2,imvtx2,level+1
798                                ,ind1,tp1r,chon1);break;
799
800          case 2 : /** l2m2=0;lm12!=0;l2m1=0;      **/
801                   mprec2d1(imvtx2,imvtx1,ivtx2,level+1
802                                ,n2ind,1,NEW);break;
803
804          case 3 : /** l2m2=0;lm12!=0;l2m1!=0;      **/
805                   mprec2d2(ivtx2,imvtx2,imvtx1,level+1
806                            ,n2ind,1,NEW,ind1,tp1r,chon1);break;
807
808          case 4 : /** l2m2!=0;lm12=0;l2m1=0;      **/
809                   mprec2d1(ivtx2,imvtx2,imvtx1,level+1
810                                ,ind2,tp2l,chon2);break;
811
812          case 5 : /** l2m2!=0;lm12=0;l2m1!=0;      **/
813                   mprec2d2(imvtx2,imvtx1,ivtx2,level+1
814                         ,ind1,tp1r,chon1,ind2,tp2l,chon2);break;
815
816          case 6 : /** l2m2!=0;lm12!=0;l2m1=0;      **/
817                   mprec2d2(imvtx1,ivtx2,imvtx2,level+1
818                            ,ind2,tp2l,chon2,n2ind,1,NEW);break;
819
820          case 7 : /** l2m2!=0;lm12!=0;l2m1!=0;      **/
821                   mprec2d3(ivtx2,imvtx2,imvtx1,level+1
822                            ,ind2,tp2l,chon2,n2ind,1,NEW,ind1,tp1r
823                            ,chon1);break;
824
825       }      /* end of switch case */
826
827       polytype3 = l3m3*4 + lm23*2 + l3m2;
828       switch ( polytype3 ) {
829          case 0 : /** l3m3=0;lm23=0;l3m2=0;      **/
830                   storepoly(ptype,ivtx3,imvtx3,imvtx2);break;
831
832          case 1 : /** l3m3=0;lm23=0;l3m2!=0;      **/
833                   mprec2d1(imvtx2,ivtx3,imvtx3,level+1
834                                ,ind2,tp2r,chon2);break;
835
836          case 2 : /** l3m3=0;lm23!=0;l3m2=0;      **/
837                   mprec2d1(imvtx3,imvtx2,ivtx3,level+1
838                                ,n3ind,1,NEW);break;
839
840          case 3 : /** l3m3=0;lm23!=0;l3m2!=0;      **/
```

```
841             mprec2d2(ivtx3,imvtx3,imvtx2,level+1
842                     ,n3ind,1,NEW,ind2,tp2r,chon2);break;
843
844          case 4 : /** l3m3!=0;lm23=0;l3m2=0;      **/
845                 mprec2d1(ivtx3,imvtx3,imvtx2,level+1
846                             ,ind3,tp3l,chon3);break;
847
848          case 5 : /** l3m3!=0;lm23=0;l3m2!=0;      **/
849                 mprec2d2(imvtx3,imvtx2,ivtx3,level+1
850                       ,ind2,tp2r,chon2,ind3,tp3l,chon3);break;
851
852          case 6 : /** l3m3!=0;lm23!=0;l3m2=0;      **/
853                 mprec2d2(imvtx2,ivtx3,imvtx3,level+1
854                          ,ind3,tp3l,chon3,n3ind,1,NEW);break;
855
856          case 7 : /** l3m3!=0;lm23!=0;l3m2!=0;      **/
857                 mprec2d3(ivtx3,imvtx3,imvtx2,level+1
858                         ,ind3,tp3l,chon3,n3ind,1,NEW,ind2,tp2r
859                         ,chon2);break;
860
861       }     /* end of switch case */
862
863       polytype4 = lm31*4 + lm12*2 + lm23;
864       switch ( polytype4 ) {
865          case 0 : /** lm31=0;lm12=0;lm23=0;      **/
866                 storepoly(ptype,imvtx1,imvtx2,imvtx3);break;
867
868          case 1 : /** lm31=0;lm12=0;lm23!=0;      **/
869                 mprec2d1(imvtx2,imvtx3,imvtx1,level+1
870                             ,n3ind,1,NEW);break;
871
872          case 2 : /** lm31=0;lm12!=0;lm23=0;      **/
873                 mprec2d1(imvtx1,imvtx2,imvtx3,level+1
874                             ,n2ind,1,NEW);break;
875
876          case 3 : /** lm31=0;lm12!=0;lm23!=0;      **/
877                 mprec2d2(imvtx3,imvtx1,imvtx2,level+1
878                         ,n2ind,1,NEW,n3ind,1,NEW);break;
879
880          case 4 : /** lm31!=0;lm12=0;lm23=0;      **/
881                 mprec2d1(imvtx3,imvtx1,imvtx2,level+1
882                             ,n1ind,1,NEW);break;
883
884          case 5 : /** lm31!=0;lm12=0;lm23!=0;      **/
885                 mprec2d2(imvtx1,imvtx2,imvtx3,level+1
886                         ,n3ind,1,NEW,n1ind,1,NEW);break;
887
888          case 6 : /** lm31!=0;lm12!=0;lm23=0;      **/
889                 mprec2d2(imvtx2,imvtx3,imvtx1,level+1
890                         ,n1ind,1,NEW,n2ind,1,NEW);break;
891
892          case 7 : /** lm31!=0;lm12!=0;lm23!=0;      **/
893                 mprec2d3(imvtx1,imvtx2,imvtx3,level+1
894                         ,n2ind,1,NEW,n3ind,1,NEW,n1ind,1,NEW);
895                 break;
896       }      /* end of switch case */
```

```
897       if (lm31 != 0) nelist[n1ind][0] = 1;
898       if (lm12 != 0) nelist[n2ind][0] = 1;
899       if (lm23 != 0) nelist[n3ind][0] = 1;
900    }       /* end of mpbm2d3 */
901
902  /**---------- CAL_MID_V ----------*/
903  long  calmidv(isvtx,ievtx,level,ind,tptr,mchon)
904    int     tptr,level;
905    int     mchon;                          /* check old or new */
906    long    isvtx,ievtx;
907    long    ind;                               /* index of edge */
908    {
909      long    ipvtx;
910      long    cecsrh();
911      void    edgevec(),V_unit(),P_prod();
912      float   unif,gau,movefac,dist;
913      float   distance(),uniform(),gauss();
914      struct vlocat  mvtx,pvtx,mnorm,pnorm,pvec,evec;
915
916    if (isvtx == 0 || ievtx == 0)
917       printf("isvtx = %d ievtx = %d mchon = %d\n"
918                                              ,isvtx,ievtx,mchon);
919    if (mchon == NEW && ind >= 0 && nelist[ind][0] == 1)
920      {if (nelist[ind][tptr] == 0)
921          printf("acess nelist = ind %d  tptr %d\n",ind,tptr);}
922
923    if (mchon == NEW)
924      {if (ind >= 0 && nelist[ind][0] == 1)
925          return(nelist[ind][tptr]);
926      }
927    else if (ind >= 0 && emlist[ind][0] == 1)
928            return(emlist[ind][tptr]);
929       dist = distance(isvtx,ievtx);
930      edgevec(&evec,isvtx,ievtx);
931       V_unit(&evec);                  /** unit vector */
932                                    /** init-locat of mid-vtx **/
933       mvtx.x = 0.5 * (varr[isvtx].x + varr[ievtx].x);
934       mvtx.y = 0.5 * (varr[isvtx].y + varr[ievtx].y);
935       mvtx.z = 0.5 * (varr[isvtx].z + varr[ievtx].z);
936                                      /** average normal **/
937       mnorm.x = 0.5 * (vnorm[isvtx].x + vnorm[ievtx].x);
938       mnorm.y = 0.5 * (vnorm[isvtx].y + vnorm[ievtx].y);
939       mnorm.z = 0.5 * (vnorm[isvtx].z + vnorm[ievtx].z);
940                                    /**  unif = uniform();
941                                           unif = 0.;    **/
942       pnorm.x = mnorm.x;  /** + unif * pvec.x;   **/
943       pnorm.y = mnorm.y;  /** + unif * pvec.y;   **/
944       pnorm.z = mnorm.z;  /** + unif * pvec.z;   **/
945
946    if (mchon == OLD)
947       if ( cesrch(ind) == 1 ) P_prod(&pnorm,evec,mnorm);
948
949       V_unit(&pnorm);                 /** vector unit-normal */
950       movefac = fixvl * dist;
951       gau = delta[level] * gauss()* movefac;
952                                       /* move midvtx location */
```

```
953       pvtx.x = mvtx.x + pnorm.x * gau;
954       pvtx.y = mvtx.y + pnorm.y * gau;
955       pvtx.z = mvtx.z + pnorm.z * gau;
956       varr[ npts ].x = pvtx.x;
957       varr[ npts ].y = pvtx.y;
958       varr[ npts ].z = pvtx.z;
959       vnorm[ npts ].x = pnorm.x;
960       vnorm[ npts ].y = pnorm.y;
961       vnorm[ npts ].z = pnorm.z;
962       ipvtx = npts; npts = npts + 1;
963    if (mchon == NEW)
964      {if (ind >= 0 && nelist[ind][0] == 0)
965          nelist[ind][tptr]=ipvtx;
966      }
967    else if (ind >= 0 && emlist[ind][0] == 0)
968            emlist[ind][tptr]=ipvtx;
969    if (mchon == NEW && ind >= 0 && nelist[ind][0] == 0)
970      {if (nelist[ind][tptr] == 0)
971          printf("store nelist = ind %d  tptr %d\n",ind,tptr);}
972      return(ipvtx);
973    }
974
975  /**- Coastline-edges searching -------------------------**/
976  long cesrch(index)
977     long   index;
978     { long  i=0;
979
980       while (i < celtc) {
981          if (ceindlt[ i ] == emltind[index]) return(1);
982          i = i + 1;
983       }     /* end of while loop */
984       return(0);
985     }
986
987  /**- EDGE_VEC ------------------------------------------**/
988  edgevec(vec,isvtx,ievtx)
989    long  isvtx,ievtx;
990    struct vlocat  *vec;
991    {  struct vlocat  svtx,evtx;
992
993       svtx.x = varr[ isvtx ].x;
994       svtx.y = varr[ isvtx ].y;
995       svtx.z = varr[ isvtx ].z;
996       evtx.x = varr[ ievtx ].x;
997       evtx.y = varr[ ievtx ].y;
998       evtx.z = varr[ ievtx ].z;
999       vec->x = evtx.x - svtx.x;
1000      vec->y = evtx.y - svtx.y;
1001      vec->z = evtx.z - svtx.z;
1002    } /* end of edgevec   */
1003
1004 /**------- P_PROD ---------------------------------------**/
1005 P_prod(norm,vt1,vt2)                /* vector cross-product */
1006      struct { float x,y,z; } *norm,vt1,vt2;
1007 {
1008      norm->x = vt1.y*vt2.z - vt1.z*vt2.y;
```

```
1009        norm->y = vt1.z*vt2.x - vt1.x*vt2.z;
1010        norm->z = vt1.x*vt2.y - vt1.y*vt2.x;
1011    }
1012
1013    /**------------------------- V_UNIT --------------------- */
1014    V_unit(vec)                 /* vector unit-normal */
1015        struct { float x,y,z; } *vec;
1016    {
1017        struct { float x,y,z; } tunit;
1018        float    length;
1019
1020        tunit.x = vec->x;
1021        tunit.y = vec->y;
1022        tunit.z = vec->z;
1023     length=(float)sqrt(sqr(tunit.x)+sqr(tunit.y)+sqr(tunit.z));
1024        vec->x = tunit.x / length;
1025        vec->y = tunit.y / length;
1026        vec->z = tunit.z / length;
1027    }
1028
1029    /**------------------------ DISTANCE --------------------- */
1030    float distance(isvtx,ievtx)
1031     long isvtx,ievtx;
1032     {  struct vlocat  svtx,evtx;
1033        float  dx,dy,dz,dist;
1034
1035        svtx.x = varr[ isvtx ].x;
1036        svtx.y = varr[ isvtx ].y;
1037        svtx.z = varr[ isvtx ].z;
1038        evtx.x = varr[ ievtx ].x;
1039        evtx.y = varr[ ievtx ].y;
1040        evtx.z = varr[ ievtx ].z;
1041        dx = evtx.x - svtx.x;
1042        dy = evtx.y - svtx.y;
1043        dz = evtx.z - svtx.z;
1044      dist=(float)sqrt(sqr(dx)+sqr(dy)+sqr(dz));
1045        return(dist);
1046     } /* end of distance   */
1047
1048    /**------------------------ LENGTH ---------------------- */
1049    int length(isvtx,ievtx)
1050     long isvtx,ievtx;
1051     {
1052        float  dx,dy,dz,dist;
1053        struct vlocat  svtx,evtx;
1054
1055        svtx.x = varr[ isvtx ].x;
1056        svtx.y = varr[ isvtx ].y;
1057        svtx.z = varr[ isvtx ].z;
1058        evtx.x = varr[ ievtx ].x;
1059        evtx.y = varr[ ievtx ].y;
1060        evtx.z = varr[ ievtx ].z;
1061        dx = evtx.x - svtx.x;
1062        dy = evtx.y - svtx.y;
1063        dz = evtx.z - svtx.z;
1064        dist=(float)sqrt(sqr(dx)+sqr(dy)+sqr(dz));
```

```
1065                        /* distance between svtx and evtx  */
1066      if (dist < epsilon) return(0);
1067         return(1);
1068    } /* end of length     */
1069
1070  /**-STORE POLYGON routine ------------------------------**/
1071  void storepoly(ptype,vtx1,vtx2,vtx3)
1072   long  vtx1,vtx2,vtx3;
1073   int   ptype;
1074   {
1075     polyarr[ polyc ][0] = 3;
1076     polyarr[ polyc ][1] = vtx1;
1077     polyarr[ polyc ][2] = vtx2;
1078     polyarr[ polyc ][3] = vtx3;
1079     plytype[ polyc ] = ptype;
1080     polyc = polyc + 1;
1081   }          /* end of storepoly */
1082
1083  /**- UINFORM DISTRIBUTION routine-----------------------**/
1084  float uniform()
1085    {
1086      float  sum;
1087      int    i;
1088      int    nrand = 2;
1089
1090      sum = 0.;
1091      for (i = 0;i < nrand;i++)
1092          sum = sum + rand() / arand - 0.5;
1093      return(sum / nrand);
1094    }        /* end of uniform */
1095
1096  /**-INITGAUSS routine-------------------------------------**/
1097   initseed(seed)
1098     int seed;
1099     {
1100      arand = pow(2.,POWER) - 1.;
1101      gfac = sqrt(12.* Nrand) / arand;
1102      srand(seed);
1103     }     /* end of initseed */
1104
1105  /**- GAUSS routine-----------------------------------------**/
1106  float  gauss()
1107     {
1108      float  sum;
1109      int    i;
1110
1111      sum = 0.;
1112      for (i = 0;i < Nrand;i++)
1113          sum = sum + (rand() - 0.5*arand)/Nrand;
1114       /** return(fabs(gfac * sum));  **/
1115      return(gfac * sum);
1116     }      /* end of gauss routine */
1117
1118  /**- fractal datafile output routine -------------------**/
1119  void fraout(str)
1120    char str[];
```

```
1121   {
1122     FILE  *output;
1123     long  i;
1124     char  filename[512];
1125
1126     strcpy(filename,str);
1127     strcat(filename,".epf");  /*  open *.fra file  */
1128     output = fopen(filename,"w");
1129
1130     fprintf(output,"%s %d %d \n","data",npts - 1,npolys);
1131     printf("%s %d %d \n","data",npts - 1,npolys);
1132                              /* write all points *.x *.y *.z */
1133     for (i = 1;i < npts; i++)
1134       fprintf(output,"%f %f %f \n"
1135                    ,varr[i].x,varr[i].y,varr[i].z);
1136
1137     for (i = 0; i < npolys; i++)              /* do polygons */
1138      { long  j,k;
1139        fprintf(output," %d",polyarr[i][0]);
1140        for (j = 1;j <= polyarr[i][0];j++)
1141            fprintf(output," %d",polyarr[i][j]);
1142        fprintf(output," \n");
1143      }      /* end of for loop to write polygon */
1144   fclose(output);
1145   }        /* end of fraout */
1146
1147  /*345678901234567890123456789012345678901234567890123456789012345678*/
1148
```

實驗結果均置於 SUN 4 workstation 中 /usr/local/SUN/skc 目錄下，而 /usr/local/SUN 為 SUN4 開給 VAX-11 的 port 故結果可直接在 VAX-11 中讀到實驗結果，同時可由 psout 輸出實驗結果。此目錄之內容如後所附。

依前述各程式目錄所產生的實驗結果可分類為

ellip ＼存放橢球的原始物及結果 (含 *.pf 及 *.ras)

mtai ＼ 〃 一維台灣碎形處理之 rasterfile (polygon)

pledge＼ 〃 一維線段台灣碎形 〃 (edge)

pmpoly ＼ 〃 二維台灣碎形處理 〃。

sphere ＼ 〃 球體之原始形狀及碎形化後之形狀
。

torus ＼ 〃 甜甜圈 〃 。

```
total 51
drwxr-xr-x  2 ps300      512 Jun 26 21:08 ellip
drwxr-xr-x  2 ps300      512 May 27 13:07 mtai
drwxr-xr-x  2 ps300      512 Jun 17 10:26 pledge
-rw-r--r--  1 ps300    45966 Jun 16 15:04 pm2d616.ln
drwxr-xr-x  2 ps300      512 Jun 20 17:13 pmpoly
drwxr-xr-x  2 ps300      512 Jun 26 21:11 sphere
drwxr-xr-x  2 ps300      512 Jun 26 21:09 torus

skc/ellip:
total 695
-rw-r--r--  1 ps300   194119 Jun 26 21:08 nellip.pf
-rw-r--r--  1 ps300   230400 Jun 26 21:08 nellip.ras
-rw-r--r--  1 ps300    14931 Jun 26 21:08 oellip.pf
-rw-r--r--  1 ps300   230400 Jun 26 21:08 oellip.ras

skc/mtai:
total 480
-rw-r--r--  1 ps300   230400 Jun 26 21:03 ctaim.ras
-rw-r--r--  1 ps300   230400 Jun 26 21:01 ptaim.ras

skc/pledge:
total 2772
-rw-r--r--  1 ps300    11795 Jun 17 10:25 cnemld.c
-rw-r--r--  1 ps300   921600 May 24 13:10 ctai3.ras
-rw-r--r--  1 ps300    10678 Jun 17 10:24 emidld.c
-rw-r--r--  1 ps300   921600 May 23 01:44 ftai3.ras
-rw-r--r--  1 ps300    12939 Jun 17 10:26 pnemld.c
-rw-r--r--  1 ps300   921600 May 23 01:37 ptai3.ras

skc/pmpoly:
total 1920
-rw-r--r--  1 ps300   175968 Jun 20 17:13 ctrim.pf
-rw-r--r--  1 ps300   230400 Jun 28 09:37 ctrim.ras
-rw-r--r--  1 ps300   230400 May 25 13:32 ctrim3.ras
-rw-r--r--  1 ps300   171993 May 27 12:52 ctrim5.pf
-rw-r--r--  1 ps300   230400 May 27 15:42 ptrim.ras
-rw-r--r--  1 ps300   230400 May 25 13:21 ptrim3.ras
-rw-r--r--  1 ps300   171993 May 27 12:51 ptrim5.pf
-rw-r--r--  1 ps300   178068 May 27 15:40 ptrim5g.pf
-rw-r--r--  1 ps300   230400 Jun 17 11:08 tptrim.ras

skc/sphere:
total 734
-rw-r--r--  1 ps300   236994 Jun 26 21:09 nsphere.pf
-rw-r--r--  1 ps300   235200 Jun 26 21:09 nsphere.ras
-rw-r--r--  1 ps300    14306 Jun 26 21:09 osphere.pf
-rw-r--r--  1 ps300   235200 Jun 26 21:09 osphere.ras

skc/torus:
total 649
-rw-r--r--  1 ps300   143375 Jun 26 21:09 ntorus.pf
-rw-r--r--  1 ps300   230400 Jun 26 21:09 ntorus.ras
-rw-r--r--  1 ps300    16391 Jun 26 21:09 otorus.pf
-rw-r--r--  1 ps300   230400 Jun 26 21:09 otorus.ras
```

以上所提之程式及使用到之相關工具程式均置於 SUN4 workstation 中 /usr/local/graphics/ps300/student/skc 目錄下。此目錄的內容隨後附上。

aniskc ＼ 存放王奧華同學所修改之 Animator 系統，用以處理吾人所造出之碎形化物體。

check ＼ 有 chedge (checkedges) chpoly (check polygon) chvtx (check vertex) 幫助 users 檢查造物體時之錯誤。

d2frat ＼ 存放所有二維碎形化的程式檔。

ellip ＼ 處理之原始物體為橢球。

p1poly ＼ 一維碎形化處理

d1frat ＼ 處理一維碎形化的程式均置於此。

stai ＼ 產生一維碎形化的原始粗略台灣

ftai ＼ 由 fixed edge normal 所處理後產生之碎形台灣。

ctai ＼ 由 current edge normal 〃 。

ptai ＼ 由 perturbating current edge normal 〃 。

mtai ＼ mtai.c 為以端点平均法向量為中心向量之處理程式。

＼ ctaim 以 current normal 方式移動。

＼ ptaim " perturbating current normal " 。

pf-std ＼ 處理 vector 結果檔及 raster 結果檔之轉換題。

pmpoly ＼ 二維碎形化處理

pmadtai.c 以端点平均法向量為中心向量處理立體物體（台灣地形）的碎形化。採用下列二种方式。

＼ ctrim ＼ 以 current normal 方式移动。

＼ ptrim ＼ 以 perturbating current normal 方式移動。

＼ data ＼ 存放台灣地形的粗略輸入資料。

sphere ＼ 處理之始物體為球體。

torus ＼ " " 甜甜圈。

utils ＼ 存放處理 Animator 所需之中間檔的工具程式。

```
total 11
drwxr-xr-x  2 ps300      1024 May 23 17:42 aniskc
drwxr-xr-x  2 ps300       512 May 19 00:03 check
drwxr-xr-x  3 ps300       512 May 26 16:47 d2frat
drwxr-xr-x  2 ps300       512 May 27 14:21 ellip
drwxr-xr-x  8 ps300       512 May 24 13:44 plpoly
drwxr-xr-x  2 ps300       512 May 19 03:25 pf-std
drwxr-xr-x  5 ps300       512 Jun 28 10:36 pmpoly
drwxr-xr-x  2 ps300       512 Jun  5 14:30 sphere
drwxr-xr-x  2 ps300      1024 May 27 02:57 src
drwxr-xr-x  2 ps300       512 May 27 15:27 torus
drwxr-xr-x  2 ps300       512 Jun 29 22:15 utils

skc/aniskc:
total 1272
-rw-r--r--  1 ps300      1141 May 15 20:05 allppl.c
-rw-r--r--  1 ps300       393 May 15 20:08 allppl.o
-rwxr-xr-x  1 ps300    147456 May  4 14:29 ani
-rwxr-xr-x  1 ps300    147456 May 15 18:42 ani2
-rwxr-xr-x  1 ps300    122880 May 17 16:09 ani320
-rwxr-xr-x  1 ps300    139264 May 15 19:41 ani640
-rwxr-xr-x  1 ps300    147456 May  4 14:29 animator
-rw-r--r--  1 ps300     33160 May 15 20:04 animator.c
-rw-r--r--  1 ps300      2569 May 15 20:04 animator.h
-rw-r--r--  1 ps300     26580 May 15 20:04 animator.o
-rw-r--r--  1 ps300      4515 May 15 20:03 driver.c
-rw-r--r--  1 ps300      5436 May 15 20:03 driver.o
-rw-r--r--  1 ps300       238 May 15 20:09 error.o
-rw-r--r--  1 ps300      4443 May 15 20:10 maingolf1.c
-rwxr-xr-x  1 ps300    122880 May 23 17:42 newani320
-rw-r--r--  1 ps300     33552 May 15 20:04 obj_sort.c
-rw-r--r--  1 ps300     15623 May 15 20:04 obj_sort.o
-rw-r--r--  1 ps300      1446 May 23 15:07 rgbhsv.c
-rw-r--r--  1 ps300      2695 May 23 15:07 rgbhsv.o
-rwxr-xr-x  1 ps300    122880 May 23 17:42 skcani
-rw-r--r--  1 ps300      3519 May 23 17:41 skccolor.c
-rw-r--r--  1 ps300      3351 May 23 17:41 skccolor.o
-rw-r--r--  1 ps300       310 May 23 17:41 skcmake
-rw-r--r--  1 ps300      5162 May 15 20:20 skcppl.c
-rw-r--r--  1 ps300      2440 May 15 20:20 skcppl.o
-rw-r--r--  1 ps300     34084 May 15 20:04 skcrender.c
-rw-r--r--  1 ps300     33190 May 15 20:04 skcrender.o
-rw-r--r--  1 ps300      3686 May 15 20:08 task_master.c
-rw-r--r--  1 ps300      2654 May 15 20:08 task_master.o
-rw-r--r--  1 ps300      4831 May 15 20:08 zdriver.c
-rw-r--r--  1 ps300      4655 May 15 20:08 zdriver.o
-rw-r--r--  1 ps300     11951 May 17 16:07 zppl.c
-rw-r--r--  1 ps300      8486 May 17 16:08 zppl.o
-rw-r--r--  1 ps300     16499 May 15 20:02 ztiler.c
-rw-r--r--  1 ps300     12101 May 15 20:02 ztiler.o

skc/check:
total 344
-rwxr-xr-x  1 ps300     24576 May  7 01:03 chedge
-rwxr-xr-x  1 ps300     24576 May  7 00:54 chpoly
-rwxr-xr-x  1 ps300     24576 May  7 00:50 chvtx
```

```
-rw-r--r--  1 ps300       7607 May 19 00:03 trim.che
-rw-r--r--  1 ps300     201447 May 18 23:47 trim.chk
-rw-r--r--  1 ps300          0 May 18 23:55 trim.perr
-rw-r--r--  1 ps300      57096 May 18 23:56 trim.vtc

skc/d2frat:
total 215
drwxr-xr-x  2 ps300        512 May 19 21:46 makepm2d
-rw-r--r--  1 ps300      32280 May  4 14:59 pm2d504.c
-rw-r--r--  1 ps300      34486 May  8 21:15 pm2d508.c
-rw-r--r--  1 ps300      41456 May 19 15:19 pm2d517.ln
-rw-r--r--  1 ps300      35840 May 19 21:41 pm2d519.c
-rw-r--r--  1 ps300      36535 May 22 15:07 pm2d522.c
-rw-r--r--  1 ps300      36319 May 26 16:47 pm2d526r.c

skc/d2frat/makepm2d:
total 35
-rw-r--r--  1 ps300      35840 May 19 21:46 makepm2d

skc/ellip:
total 229
-rw-r--r--  1 ps300       5384 May 27 14:10 ellip.det
-rw-r--r--  1 ps300     181075 May 19 21:51 ellip.epf
-rw-r--r--  1 ps300        304 May 27 14:20 ellip.obj
-rw-r--r--  1 ps300         98 May 19 21:28 ellip.par
-rw-r--r--  1 ps300        349 Jun 26 21:52 ellip.scn
-rw-r--r--  1 ps300      10185 May 19 21:29 ellip.std
-rw-r--r--  1 ps300      18146 May 19 21:51 t.er

skc/p1poly:
total 30
drwxr-xr-x  2 ps300        512 May 24 13:43 ctai
drwxr-xr-x  2 ps300        512 Jun 21 15:38 d1frat
drwxr-xr-x  2 ps300        512 May 24 13:43 ftai
-rwxr-xr-x  1 ps300      24576 May 23 01:31 make_bin
drwxr-xr-x  4 ps300        512 Jun 28 02:33 mtai
drwxr-xr-x  2 ps300        512 May 24 13:43 ptai
drwxr-xr-x  2 ps300        512 May 24 13:43 stai

skc/p1poly/ctai:
total 64
-rwxr-xr-x  1 ps300      40960 Mar 28 18:23 cnem1d
-rw-r--r--  1 ps300      11795 May 22 16:49 cnem1d.c
-rw-r--r--  1 ps300       7993 May 24 13:21 ctai.epf
-rw-r--r--  1 ps300        166 May 22 21:03 ctai.obj
-rw-r--r--  1 ps300         25 May 24 13:21 ctai.par
-rw-r--r--  1 ps300        398 May 24 12:47 ctai.scn
-rw-r--r--  1 ps300        604 May 23 10:00 ctai.std

skc/p1poly/d1frat:
total 113
-rw-r--r--  1 ps300      11795 Jun 17 10:41 cnem1d.c
-rw-r--r--  1 ps300      11917 Jun 20 11:04 cpnem1d.c
-rw-r--r--  1 ps300      15731 Jun 21 15:32 cpnem1d.ln
-rw-r--r--  1 ps300      12854 Jun 21 15:30 cpnem1d.tmp
-rw-r--r--  1 ps300      10678 Jun 19 11:20 fnem1d.c
```

```
-rw-r--r--  1 ps300       13731 Jun 21 12:23 fnem1d.ln
-rw-r--r--  1 ps300       11106 Jun 21 12:21 fnem1d.tmp
-rwxr-xr-x  1 ps300        1422 May 23 01:30 make_bin.c
-rw-r--r--  1 ps300        8310 Mar 14 22:47 mid1d.c
-rw-r--r--  1 ps300       12939 Jun 17 10:41 pnem1d.c

skc/p1poly/ftai:
total 64
-rwxr-xr-x  1 ps300       40960 Mar 28 16:38 emid1d
-rw-r--r--  1 ps300       10678 May 22 16:48 emid1d.c
-rw-r--r--  1 ps300        8907 May 24 13:24 ftai.epf
-rw-r--r--  1 ps300         159 May 22 21:04 ftai.obj
-rw-r--r--  1 ps300          25 May 24 13:24 ftai.par
-rw-r--r--  1 ps300         276 May 23 10:04 ftai.scn
-rw-r--r--  1 ps300         605 May 23 01:31 ftai.std

skc/p1poly/mtai:
total 66
drwxr-xr-x  2 ps300         512 Jun 28 01:54 ctaim
-rwxr-xr-x  1 ps300       24576 May 25 00:58 makem_bin
-rw-r--r--  1 ps300       17692 Jun 28 01:55 mtai.c
-rw-r--r--  1 ps300       21640 Jun 28 01:56 mtai.ln
drwxr-xr-x  2 ps300         512 Jun 28 01:55 ptaim

skc/p1poly/mtai/ctaim:
total 170
-rwxr-xr-x  1 ps300       49152 Jun 28 01:54 ctaim
-rw-r--r--  1 ps300       35276 Jun 26 21:02 ctaim.det
-rw-r--r--  1 ps300       79020 Jun 26 21:01 ctaim.epf
-rw-r--r--  1 ps300         167 May 25 01:00 ctaim.obj
-rw-r--r--  1 ps300          74 Jun 26 20:43 ctaim.par
-rw-r--r--  1 ps300         389 Jun 24 09:39 ctaim.scn
-rw-r--r--  1 ps300        3598 Jun 26 20:40 ctaim.std
-rw-r--r--  1 ps300        1382 Jun 26 21:01 t.er

skc/p1poly/mtai/ptaim:
total 133
-rwxr-xr-x  1 ps300       49152 Jun 28 01:55 ptaim
-rw-r--r--  1 ps300       24088 Jun 26 21:00 ptaim.det
-rw-r--r--  1 ps300       53064 Jun 26 20:41 ptaim.epf
-rw-r--r--  1 ps300         167 May 25 01:21 ptaim.obj
-rw-r--r--  1 ps300          74 May 27 02:37 ptaim.par
-rw-r--r--  1 ps300         389 Jun 24 09:40 ptaim.scn
-rw-r--r--  1 ps300        3598 Jun 26 20:41 ptaim.std
-rw-r--r--  1 ps300        1382 Jun 26 20:41 t.er

skc/p1poly/ptai:
total 65
-rwxr-xr-x  1 ps300       40960 Mar 30 10:27 pnem1d
-rw-r--r--  1 ps300       12939 May 22 16:49 pnem1d.c
-rw-r--r--  1 ps300        7921 May 24 13:42 ptai.epf
-rw-r--r--  1 ps300         159 May 22 17:34 ptai.obj
-rw-r--r--  1 ps300          25 May 24 13:42 ptai.par
-rw-r--r--  1 ps300         404 May 24 12:42 ptai.scn
-rw-r--r--  1 ps300         605 May 23 01:35 ptai.std
```

```
skc/plpoly/stai:
total 4
-rw-r--r--  1 ps300         159 May 22 16:50 stai.obj
-rw-r--r--  1 ps300          25 May 22 16:44 stai.par
-rw-r--r--  1 ps300         276 May 23 09:48 stai.scn
-rw-r--r--  1 ps300         604 May 23 09:52 stai.std

skc/pf-std:
total 72
-rwxr-xr-x  1 ps300       24576 May  9 16:55 echvtx
-rwxr-xr-x  1 ps300       24576 Apr 26 17:31 make_pf
-rwxr-xr-x  1 ps300       24576 Apr 29 18:54 mk-std

skc/pmpoly:
total 242
-rwxr-xr-x  1 ps300       57344 Jun 28 03:25 a.out
-rwxr-xr-x  1 ps300       57344 Jun 26 21:32 cnem2d
drwxr-xr-x  2 ps300         512 Jun 26 21:44 ctrim
drwxr-xr-x  2 ps300         512 May 24 04:08 data
-rw-r--r--  1 ps300       47824 Jun 28 10:36 pm2d.ln
-rw-r--r--  1 ps300       39788 Jun 28 10:35 pm2d.tmp
-rw-r--r--  1 ps300       41145 Jun 28 10:24 pm2dtai.c
drwxr-xr-x  2 ps300         512 May 27 15:32 ptrim

skc/pmpoly/ctrim:
total 817
-rw-r--r--  1 ps300       70352 Jun 26 21:44 ctrim.det
-rw-r--r--  1 ps300      162668 Jun 26 21:43 ctrim.epf
-rw-r--r--  1 ps300         305 Jun 20 15:04 ctrim.obj
-rw-r--r--  1 ps300          97 Jun 20 13:25 ctrim.par
-rw-r--r--  1 ps300      171627 May 22 16:23 ctrim.pf
-rw-r--r--  1 ps300         419 Jun 26 21:57 ctrim.scn
-rw-r--r--  1 ps300       28331 May 22 15:35 ctrim.std
-rw-r--r--  1 ps300      160635 May 24 11:34 ctrim3.epf
-rw-r--r--  1 ps300      157371 May 24 15:21 ctrim7.epf
-rw-r--r--  1 ps300       36942 Jun 26 21:43 t.er

skc/pmpoly/data:
total 157
-rw-r--r--  1 ps300       32532 May 24 04:08 tai.ln
-rw-r--r--  1 ps300       19689 May 22 14:58 tai367.608
-rw-r--r--  1 ps300       28331 May 22 14:59 tai494.919
-rw-r--r--  1 ps300          47 May 24 04:08 trim.cpar
-rw-r--r--  1 ps300       23792 May 24 04:08 trim.dat
-rw-r--r--  1 ps300       28331 May 24 04:08 trim.scl
-rw-r--r--  1 ps300       22634 May 24 04:08 trim.sor
-rw-r--r--  1 ps300          22 May 24 04:08 trim.spar

skc/pmpoly/ptrim:
total 649
-rw-r--r--  1 ps300       71732 May 27 15:32 ptrim.det
-rw-r--r--  1 ps300      166046 May 27 13:53 ptrim.epf
-rw-r--r--  1 ps300         304 May 24 15:21 ptrim.obj
-rw-r--r--  1 ps300          97 May 27 03:07 ptrim.par
-rw-r--r--  1 ps300         421 Jun 26 21:35 ptrim.scn
-rw-r--r--  1 ps300       28331 May 22 15:34 ptrim.std
```

```
-rw-r--r--  1 ps300      160916 May 24 11:38 ptrim3.epf
-rw-r--r--  1 ps300      157236 May 24 15:18 ptrim7.epf
-rw-r--r--  1 ps300       34828 May 27 13:53 t.er

skc/sphere:
total 358
-rw-r--r--  1 ps300       89360 May 19 21:57 sphere.det
-rw-r--r--  1 ps300      210145 May 19 21:55 sphere.epf
-rw-r--r--  1 ps300         197 Jun  5 14:34 sphere.obj
-rw-r--r--  1 ps300          98 May 19 21:32 sphere.par
-rw-r--r--  1 ps300       14255 Apr 29 16:18 sphere.pf
-rw-r--r--  1 ps300         352 Jun 26 21:52 sphere.scn
-rw-r--r--  1 ps300       13183 May 19 21:31 sphere.std
-rw-r--r--  1 ps300       16930 May 19 21:55 t.er
-rw-r--r--  1 ps300        6304 Jun  5 14:30 tsphere.det

skc/src:
total 365
-rw-r--r--  1 ps300         240 Feb  1 15:25 Makefile
-rwxr-xr-x  1 ps300      114688 May 27 02:57 animator
-rw-r--r--  1 ps300       33013 Feb  2 16:00 animator.c
-rw-r--r--  1 ps300        2563 Feb  1 15:22 animator.h
-rw-r--r--  1 ps300       26744 May  8 18:03 animator.o
-rw-r--r--  1 ps300        4493 May 27 02:53 driver.c
-rw-r--r--  1 ps300        4344 May 27 02:57 driver.o
-rw-r--r--  1 ps300         167 Feb  1 15:22 error.c
-rw-r--r--  1 ps300         238 May  8 18:06 error.o
-rw-r--r--  1 ps300       33538 May 27 02:54 obj_sort.c
-rw-r--r--  1 ps300       15623 May 27 02:55 obj_sort.o
-rw-r--r--  1 ps300       11318 May 27 02:53 pixels.c
-rw-r--r--  1 ps300        8222 May 27 02:56 pixels.o
-rw-r--r--  1 ps300       31905 May 27 02:51 render.c
-rw-r--r--  1 ps300       31267 May 27 02:56 render.o
-rw-r--r--  1 ps300        3497 Feb 14 17:30 task_master.c
-rw-r--r--  1 ps300        2553 May  8 18:04 task_master.o
-rw-r--r--  1 ps300       16398 May 27 02:52 tiler.c
-rw-r--r--  1 ps300       11973 May 27 02:56 tiler.o

skc/torus:
total 193
-rw-r--r--  1 ps300       19362 May 19 22:42 t.er
-rw-r--r--  1 ps300        3511 May 10 21:09 to8.sdat
-rw-r--r--  1 ps300        7176 May 27 15:27 torus.det
-rw-r--r--  1 ps300      133566 May 19 22:42 torus.epf
-rw-r--r--  1 ps300         311 May 27 14:39 torus.obj
-rw-r--r--  1 ps300          98 May 19 03:26 torus.par
-rw-r--r--  1 ps300         302 Jun 26 21:53 torus.scn
-rw-r--r--  1 ps300       15110 May 19 03:23 torus.std

skc/utils:
total 162
-rwxr-xr-x  1 ps300       24576 May  3 13:24 a.out
-rw-r--r--  1 ps300        3388 May  7 01:01 chedge.c
-rw-r--r--  1 ps300        3238 May  7 00:53 chpoly.c
-rw-r--r--  1 ps300        4532 May  7 00:50 chvtx.c
-rwxr-xr-x  1 ps300       24576 Feb 20 21:55 color_bin
```

```
-rwxr-xr-x  1 ps300        1089 Feb  2 14:44 color_bin.c
-rw-r--r--  1 ps300        3907 May  9 16:55 echvtx.c
-rwxr-xr-x  1 ps300        1419 Apr 29 17:38 make_bin.c
-rwxr-xr-x  1 ps300        1369 May  2 16:40 make_pf.c
-rwxr-xr-x  1 ps300        1207 May  9 16:23 make_std.c
-rwxr-xr-x  1 ps300       24576 Feb 17 18:19 makem_bin
-rw-r--r--  1 ps300         467 Dec 11  1988 mk-pf.c
-rw-r--r--  1 ps300         442 May  2 16:37 mk-std.c
-rw-r--r--  1 ps300         652 May 24 22:28 mtri.form
-rw-r--r--  1 ps300        4140 May 24 22:21 tai.form
-rwxr-xr-x  1 ps300       24576 May  2 22:04 zcolor
-rw-r--r--  1 ps300        4768 May  2 22:04 zcolor.c
-rwxr-xr-x  1 ps300       24576 May  1 16:39 zscale
-rw-r--r--  1 ps300        3343 May  1 16:45 zscale.c
```