序言

「我希望能讓電腦幫我畫圖」。

現今的電腦軟體中，已經有許多不錯的建模軟體可供大眾使用，從免費的 Wings 3D、有著特殊功能取向的六角大王、適合初學者的 Lightware、到進階的 MAYA、3DS-MAX 等。

由於興趣的關係，我有幸接觸到其中幾種建模軟體，能夠處理一些簡單的模型建模；對於繪畫也有一定的興趣。然而非美術科班出身的背景使得我雖能快速的建模，卻無法隨性的勾勒出具有正確的物件相對關係、具有美感的繪畫。在建模的過程中，偶然的思考到，有沒有一種技術能夠把這些 3D-Model Render 成平面的繪畫呢？

對於初學繪畫的人來說，若沒有經過反覆的素描訓練，是很難正確的掌握所繪物件彼此間的相對空間關係的。如果有某種工具，可以幫我把建出來的 3D 模型資料，轉換成平面的手繪圖，甚至藉著電腦算圖的能力，生成連續的影像，即所謂的”動畫”，或許是一件有趣的事吧？

於是我選定了這個題目作為研究所研習的方向。在研究所的兩年裡，首先我要感謝我的指導老師：陳履恆教授。陳老師並沒有強制限定我的研究方向，給了我很自由的發展空間。同時老師也經常適時的給予我研究上的方向與意見，讓我能夠順利的完成這篇研究論文。

當然也要感謝我的家人們，在這多年的求學期間，他們總是無怨無悔的為我付出並且時常關心我的生活狀況。雖然家住的近，但他們也能夠容忍我常常隔好幾個星期才回家。同時也感謝學校以及埔里的各個單位及商家，你們提供的打工機會讓我能夠在埔里自給自足。

最後，在此感謝所有我認識的同學們，你們使我的求學生活多采多姿，在我們辛苦的時候能互相扶持，一同成長。謝謝大家。

論文名稱：以模型形體為導向的非擬真筆觸成像技術
院校系：國立暨南國際大學資訊工程學系　　頁數： 59
畢業時間：九十五年八月　　學位別：碩士
研究生：蘭心皓　　指導教授：陳履恆　博士

摘要

非擬真筆觸成像技術 (NPR)在電腦圖學的領域中由來已久，相較於正常的電腦圖學—追求精確且正確的成像，NPR 著重於強調生成影像其本身的風格特性，採用誇張的手法來呈現目標本身的特性，進而呈現出各種風格的影像生成結果。在本篇論文中，我們提出一種新的 NPR 影像生成法，藉由抽取三維模型自身的特性如輪廓以及骨架結構等，生成以此為參照基礎所產生的連續筆觸影像，以及動畫。藉由我們所提出的方法，我們可以利用讀入的三維場景以及動畫資訊，生成模擬出具有強烈畫風特性如梵谷、高更等獨特性質的繪畫影像。同時，實驗結果也證實我們可以利用連續的影像，生成出連貫的繪畫風格動畫影片。

關鍵詞：筆觸、畫風、Non-Photorealistic Rendering(NPR)、模型、電腦圖學、Line-Integral-Convolution(LIC)、3D-skeletons、印象派。

Title of Thesis: Shape-Oriented Brush Stroke Synthesis in Non-Photorealistic Rendering
Name of Institute: National Chi Nan University, Dept. of CSIE Pages: 59
Graduation Time: August 2006 Degree Conferred: Master
Student Name: Jimmy Lin Advisor Name: Dr. Lieu-Hen Chen

Abstract

Non-photorealistic rendering (NPR) has been an important topic in the research field of Computer Graphics. Instead of pursuing the photo-realism of images, researchers of NPR focus on the representation of painting style, such as the exaggeration expression over the subjects and the visual effects of brush strokes. In this paper, we propose a new model-based NPR method to simulate and synthesize continuous brush strokes by utilizing the auxiliary bone structure of models and other necessary information which is extracted from the shapes of three-dimensional models. Furthermore, painting techniques of artists are considered and simulated in our system. By applying our methods, images of scenery, that consists both of moving and still objects, with the painting styles of great impressionists, C'ezanne and Van Gogh, are synthesized. The experiment results show that our method successfully generates interesting NPR animation with continuous brush strokes.

Keyword: brush stroke, paint style, Non-Photorealistic Rendering (NPR), model, computer graphics, Line-Integral-Convolution (LIC), 3D-skeletons, Impressionism.

目錄

摘要 2
目錄 4
第一章　緒論 6
1.1　研究動機與目的 6
1.2　論文編排模式 6
第二章　研究背景 8
2.1 筆觸風格的分析與量化 8
2.1.1 筆觸特性 10
2.1.2　顏色特性 10
2.2 筆觸風格列舉以及探討 12
2.2.1 形 13
2.2.2 質感的表現 14
2.2.3 地面，空間感等相對關係的表現 15
2.2.4 光影的呈現 16
第三章　系統概述 17
3.1 構想 17
3.2 系統架構 18
3.3 系統運作流程 20
第四章　資料的讀取及整理 22
4.1 X-loader 22
4.1 X-file Adapter 23
第五章　虛擬畫布: Modified Z-buffer 26
5.1 Modified Z-buffer 26
5.2 Rasterization 28
第六章　光與影的生成 33
6.1 Volume Shadow 33
第七章　虛擬畫筆: Stroke Calculator 35
6.1 Outline Vector 35
6.2 Single Direction Vector 37
6.3 2D Shape-based Vector 38
6.3 3D-Model based Vector: skeleton 40
第八章　筆觸的渲染 42
8.1 Line Integral Convolution (LIC) 42

8.2 複合圖層 43
第九章　筆觸風格 45
9.1 筆刷的選擇 45
9.2 筆觸寬度模擬 45
9.3 潦草筆觸的生成 46
第十章　動畫生成 47
第十一章　研究成果 49
第十二章　結論以及未來工作 55
12.1 結論 55
12.2 未來工作 55
參考文獻 57

第一章　緒論

1.1　研究動機與目的

在電腦圖學的領域中，對於筆觸的模擬及研究由來已久，其中對此著墨甚多的，就屬Non-Photorealistic Render(NPR)這一塊領域了。

然而，在瀏覽了幾種近年來較具指標意義的NPR成像演算法之後，仍無法找到離我們的目標較接近的成果。利用電腦模擬人繪，就彷佛是無窮的逼近般，並沒有一個完美的模擬方法。而目前所有的NPR生成，無論是鋼筆畫、素描、油畫或國畫的模擬，有些方法採用Image procession的方法，將照片生成具有NPR 風格的結果；有些雖然使用3D-model作為參考資料，但仍需要使用者互動式的指定筆觸的走向；再不就是採用局部的觀點生成筆觸方向，結果造成筆觸的不連續。這些模擬模式，無論從畫者或觀者的角度來看，都仍有相當大的進步空間。

因此，我們提出這篇研究論文，希望能夠參考3D-model的資訊，並且觀察現有畫家在繪畫時採用的幾種常用的技法，藉由這些資訊來模擬出能夠更逼近人類手繪繪畫的結果。同時，我們更希望能夠藉由這些結果，整合3D建模本身的優勢，生成連續的影像動畫。

1.2　論文編排模式

為了能夠生成”高可信度”的模擬結果，我們嘗試從現有的畫派畫家的繪畫中，抽取出可供量化的參數。並且，為了能充分模擬此種繪畫的感覺，我們將探討其上色、繪

線的幾種模式。此一部分將於稍後的第二章來做探討。

在第三章，我們講解關於此系統的設計構想、以及系統的主要架構與流程。

第四章將資料讀取時所處理的前置工作做大致上的介紹，其中包含讀取 X-file 的程式以及將其分析重整所使用的 Adapter 兩個部分。

第五章，我們講解本系統自訂的虛擬畫布：Modified Z-buffer。無論是紀錄資訊、筆觸計算或 Layer 之間的混色等皆在此一 Buffer 上處理，故此一 Z-buffer 亦可視為本系統在運作上的核心。第六章則大致說明在實作時所採用的 Shading method，以及陰影的相關運算。在上述的工作準備完成後，我們在第七章講解筆觸的走向生成。根據上述的觀察統計，共可分為 Outline Vector、Single Direction Vector、2D-based Vector 以及 3D-based Vector 共四個主要的部分。

第八章主要講解 Layer 的使用概念，以及將比觸走向統合生成筆觸所採用的 Line Integral Convolution (LIC) Method。第九章則是描述各種在研究中所使用的筆觸風格生成，包含筆刷的選擇，筆觸粗細分部、潦草的模擬等。在第十章，我們探討關於連續生成影格時，所造成的筆觸連續性問題。最後，我們在第十一章展示我們的研究成果，以及第十二章的結論說明及未來的工作。

第二章　研究背景

Non Photorealistic Rendering (NPR) 在電腦圖學中的一個特殊領域。相較於正常的電腦圖學—追求精確且正確的成像，NPR 著重於強調生成影像其本身的風格特性，採用誇張的手法來呈現目標本身的特性，進而呈現出各種風格的影像生成結果。

本節我們針對本篇研究欲達成的目標：使用 3D-model base 的 NPR 來模擬出連續且正確的畫家筆觸動畫。分別詳述研究的方向，以及探討幾種可行的方式。

2.1 筆觸風格的分析與量化

在繪畫的世界中，筆觸乃產生一張完整手繪圖的基本元件。即使是同一個欲描繪的物體，根據不同的繪畫方式，使用的筆觸亦不相同。

我們從印象畫派的幾種畫風中可以發現其中明顯的差異性：

[圖一]neo impressionism：pointillism 點描派的繪圖，將顏色還原成原本的七色，用點描繪在畫布上。

[圖二]印象派賽尚的靜物。強調繪畫的純粹性、重視繪畫的形式構成。作畫常以黑色的線勾畫物體的輪廓。

[圖三] post impressionism : Van Gogh

後印象派修正了印象派太注重光影變化以及新印象派忽略主題本身的形與色，摻入東方畫家的主觀觀察，純粹的表現眼睛所看到的形與色。其中以梵谷的畫作 – 其明顯的筆觸變化最用常來代表印象派的畫風。

由上述幾張圖我們可以發現，不同的筆觸風格所表現出來的繪畫風格亦不相同。為了提供我們更具體的研究成果比對，在前期的研究過程中，我們嘗試由以下幾種方式來探討繪畫上可量化的資訊：

(1)筆觸特性

(A) 筆觸長短的分布

(B) 筆觸寬度的描繪習慣

(C) 筆觸走向角度的分布

(2)顏色特性

(A)contour 的處理手法

(B)色塊大小的分布

(C)用色的習慣

(D)色彩對比的分布

2.1.1 筆觸特性

然而，就 Image Processing 的觀點來看，試圖從影像中擷取出條狀的筆觸資訊，並不是一件簡單的事，分布在畫布上的筆觸資訊，經過層層的疊加，並不容易區分出其原本的屬性及資訊，再加上缺乏近一步更精細的數位化畫作，使得此一部分的資訊擷取相當困難。因此，有了這一部分的結論後，我們的研究便朝向針對畫布上色彩空間分布的分析。

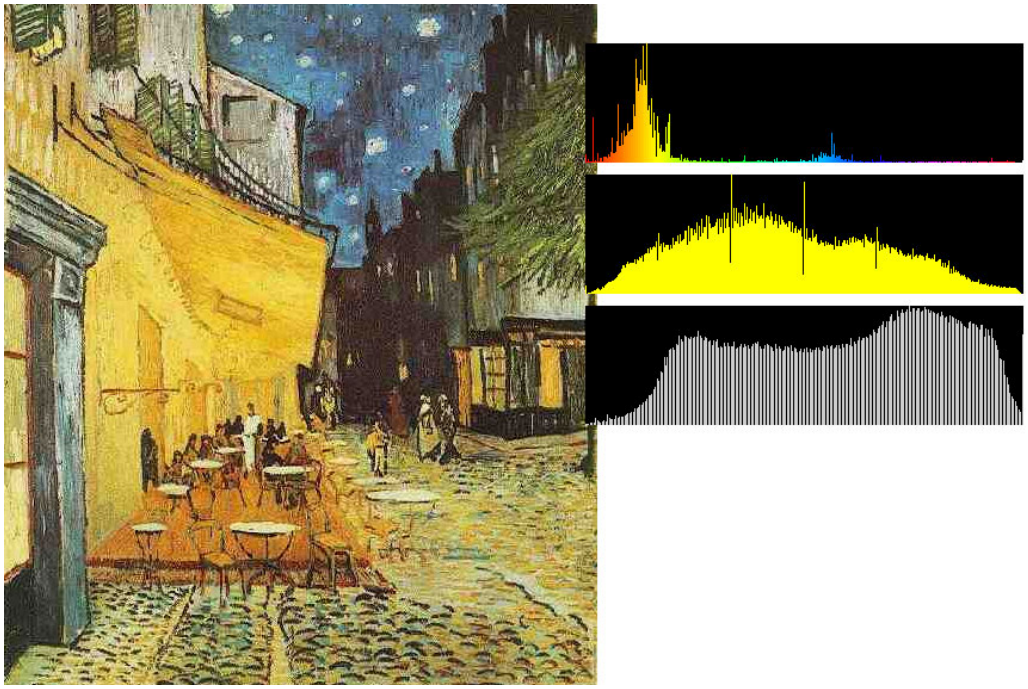
2.1.2 顏色特性

我們將繪畫點陣圖影像的 RGB color 轉換成 HSV 色彩座標系，之後分離 H,S,V 比較各畫家的用色習慣與對比關係。

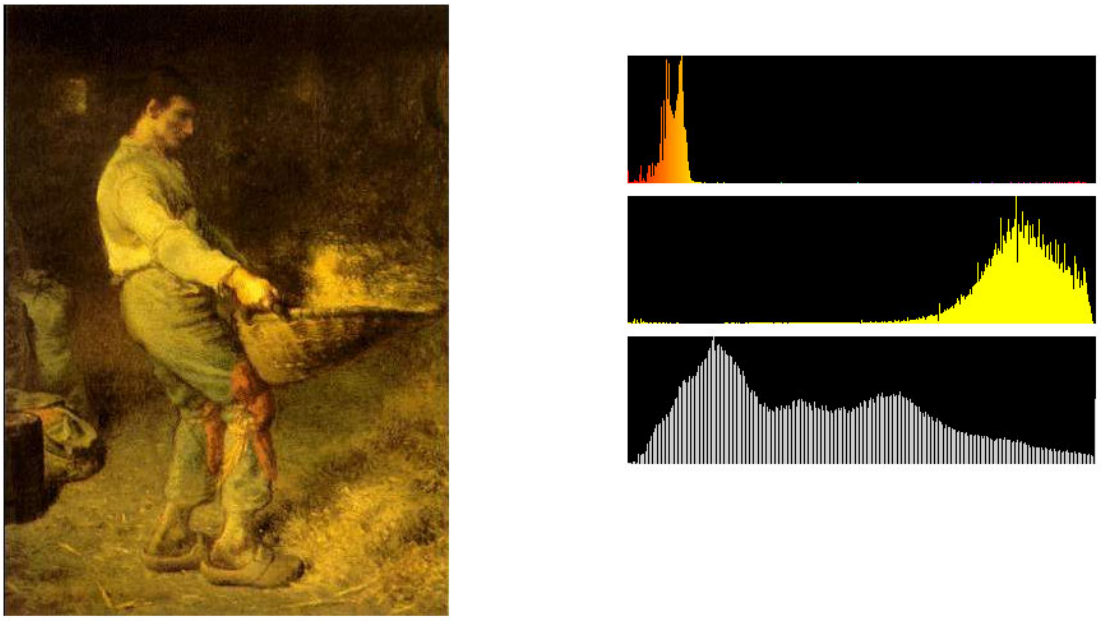
從實際的顏色分析中，我們觀察到幾個有趣的現象：

1. 梵谷、高更、米勒等著名的印象派畫家，其用色大多分布於 H1~100 之間。(白色框線部份)

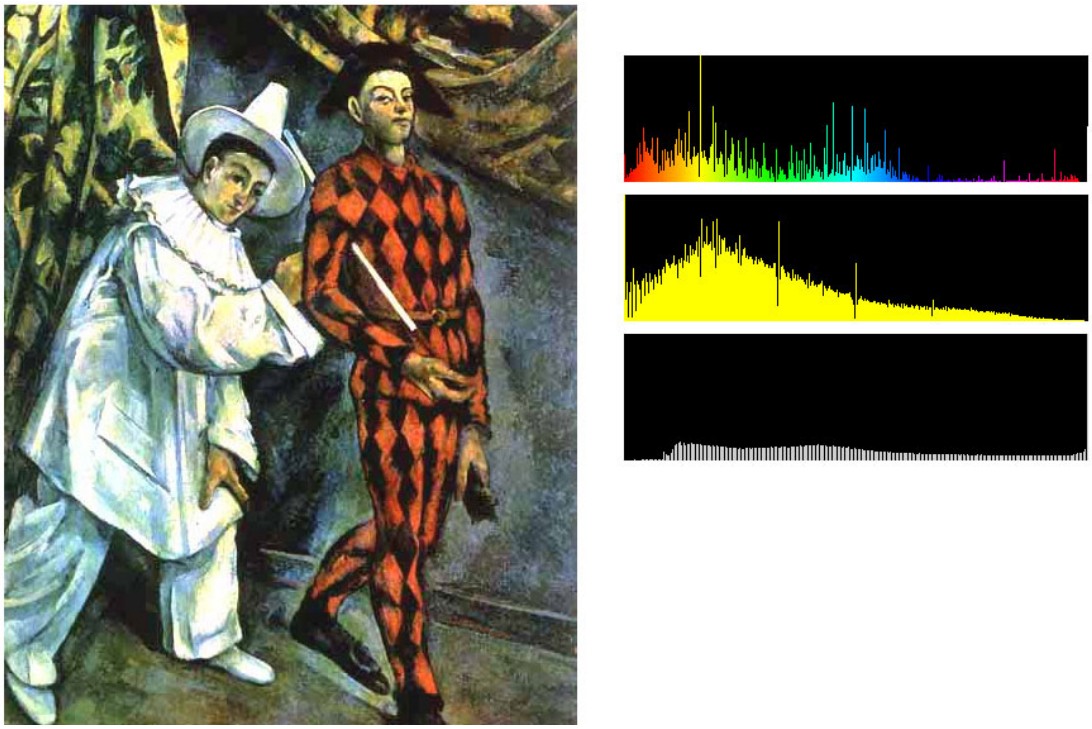
2. 細分其顏色調性分布，可以發現梵谷的繪畫，色調(S)大多分布在中間(25%~60%)，鮮少使用高色調的顏色；米勒的繪畫，色調則大多分布在頂端(60%+)，亦即偏向使用高色調的色彩來呈現其繪畫。



[圖四]梵谷畫作 - 三條分析長條圖由上至下依次為 H(Hue)、S(Saturation)、V(Value)，此圖可以發現梵谷的色相分布於 H:1~100 左右、用色飽和度則偏中間帶。



[圖五]米勒畫作 - 與梵谷的共通色相分布：H(1~100)，其用色飽和度則偏好高飽和。



[圖六]塞尚畫作 – 主要色相分布仍然相同，但是畫中常常參有藍色色系的顏色。

雖然取得的資訊有限，但這些顏色分布的資訊仍有助於我們接下來建構虛擬繪畫時相關資訊的參照。

2.2 筆觸風格列舉以及探討

為了能夠模擬出實際畫作的筆觸風格，本節我們列舉幾種在 NPR 系統中實作的筆觸風格以及成像方式。

2.2.1 形

欲描繪一個物體，正確的描繪其形體將是最重要的一個步驟。看似複雜的形體其實也可以看作簡單的幾何圖形所組成。因此在描繪形體時，不應過於拘泥細微的形狀，只要能充分表達其物體的主形體即可。

觀察右邊的圖不難發現，僅僅藉由輪廓的表現，就能概要性的將物體的三維形體呈現出來。且輪廓本身也具有粗細、濃淡等變化。

[圖七] 由簡單的輪廓線索構成的茶壺。

由於我們的參考物件為實體 3D 模型，因此整個物件的外框線(Outline)將根據生成方式區分為三種：Silhouettes, Boundaries, Discontinuities。

(A)Silhouettes 當線段的相鄰兩三角形其中一個三角形法向量與視角夾角小於 90 度，另外一個大於 90 度，即代表這條線處於一個可視的三角形與一個不可視(culled)的三角形中間。此種線段稱為 Silhouettes。

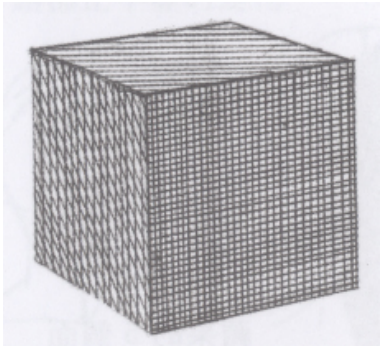
(B)Boundaries 當線段只有一個相鄰三角形時，即代表這條線在模型的邊緣，此種線段稱為 Boundaries。

(C)Discontinuities 當線段的相鄰兩三角形法向量夾角大於某個固定角度時，即判斷此線段需要繪出，稱為 Discontinuities。

2.2.2 質感的表現

大體而言，每種物體都有其相異的質感，如光滑、粗糙、笨重、細膩等，善用描繪方式的不同，可以利用濃淡的變化及筆觸的風格來達到這些題材的特色。

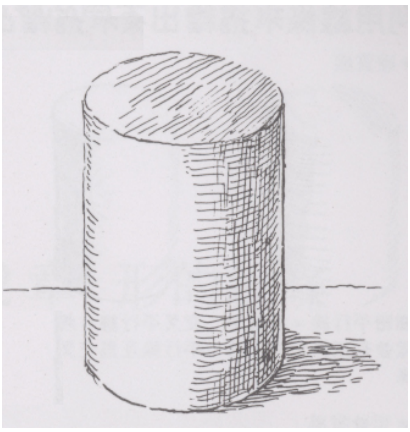
(A) 單向筆觸：此種筆觸通常用以表現平面，或是重複堆疊以呈現不同的明暗深淺。由於筆觸的走向為固定的方向，故僅需偵測其筆觸與外框線的碰撞偵測。



［圖八］單向性筆觸。

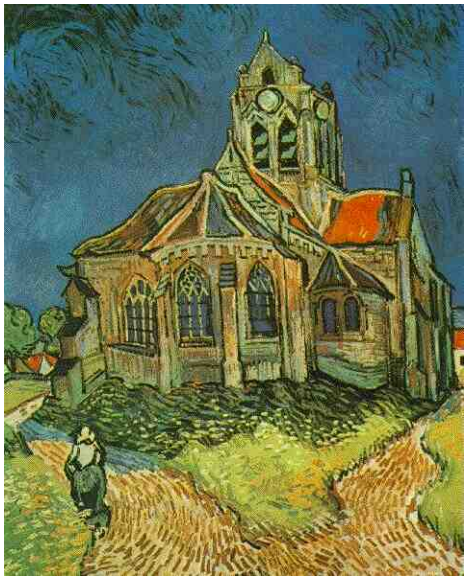
(B) 以模型形體為主的筆觸：在傳統的繪畫中，物件的形體表現一直是重要的一環。而筆觸在這個環節所扮演的角色，往往是用來強調出物件的三維特性，讓觀者能由 2D 的繪畫在腦海中重建出物件原始的 3D 型態。即是說，筆觸往往能夠幫助觀者更正確的認知一幅畫中的物件。

在本論文中我們物體本身的 Skeleton 特性來計算此一筆觸。Skeleton 是 3D-model 一項很重要的形狀特徵，此資訊容許我們根據 skeleton 重建 3D-model 的形體，也常應用在許多不同的領域。

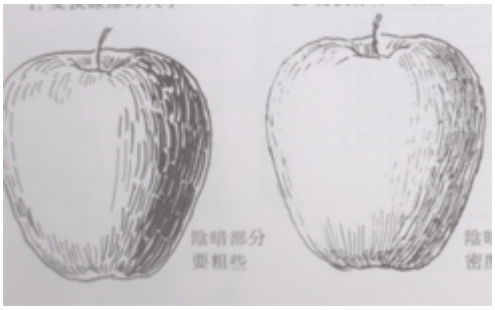


［圖九］用以描繪物件形體特性的筆觸。

(C) 根據 3Dmodel 投影後的 2D-image shape 的特性所產生的筆觸。如下圖中筆觸的走向。在繪圖的時候，有時我們需要額外的線條來強調物體本身 2D 的形狀。這種繪畫手法尤其在當物件本身具有不規律起伏的表面時。大致來說，此種繪線常會隨著物件本身輪廓線起伏、生成。



[圖十] 梵谷 – 奧維的教堂。其中部分筆觸沿著物件的形而出現圍繞的情形。



[圖十一] 鋼筆畫：物件內部的筆觸沿著物體的外形而作導向的繪製。

2.2.3 地面，空間感等相對關係的表現

在傳統的 Image-base 的 NPR 成像上，由於缺乏準確的物件－地表之間相對關係的資訊，生成的影像往往會在物件－背景 物件－物件之間的邊界出現模糊不清的問題，導致缺乏精確的相對關係。本研究採用 model-base 的生成方式，由於有切確的實體參考模型，生成的筆觸可以明確定義其所屬位置，故可以避免這部分的缺點。此外，在傳統繪畫上，精準的描繪出物件透過透視矩陣所形成的影像，亦非易事。透過 3D-model 與場景中的相對關係，我們可以藉由矩陣的運算定位出出正確的物件形體。

2.2.4 光影的呈現

在 NPR 的影像生成中，所有的像素顏色僅僅擔任原始的畫面參考色，由這些參考色中經過分層、混色以及筆觸生成之後才生成最終的輸出影像。即是說，對顏色精細度的要求較低。因此，在我們的 NPR 系統中僅採用簡單的 Flat Shading model 作為基本的 Shading 方法。

而光影的呈現向來是繪畫中重要的一環，陰影的生成由於 Flat Shading 先天上的不足，為了能夠正確的呈現陰影的位置，在研究中我們導入 Volume Shadow 的方式來計算陰影遮蓋的空間，進而計算陰影部分的筆觸。

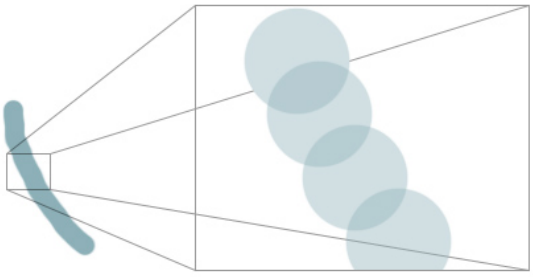
第三章　系統概述

本章我們說明我們所提出的 NPR System，包括早期的構想、系統架構以及運作流程圖。

3.1 構想

考慮最原始的筆觸構成。

假設單點的筆觸圖案為一實心同色圓，則沿某一軌跡連續貼出此圖案即可模擬出最簡單的筆觸。而根據此一基本想法加以擴充，如基本筆觸的圖形變化、分布的密度、透明度等，便產生了構成不同風格的筆觸所需的各種參數。



［圖十二］筆觸的細節。

而此一筆觸遵循的軌跡，更是構成各種風格的繪畫的一個重要因素。為了能生成第四章討論的幾種筆觸風格—與模型有關聯性的筆觸以及與外型有關聯性的筆觸，我們需要一個模擬畫布的資料結構。這張模擬畫布的每一個 Pixel，必須可以自由的參考 3Dmodel 以及矩陣運算過後的 2D-shape，或者整個世界場景中的相關資料如光影、方位等。如此才可以根據這些資訊生成可信的筆觸。

基於這個構想，我們嘗試改良原本繪圖卡所提供的 Z-buffer 這樣的繪圖機制。藉由

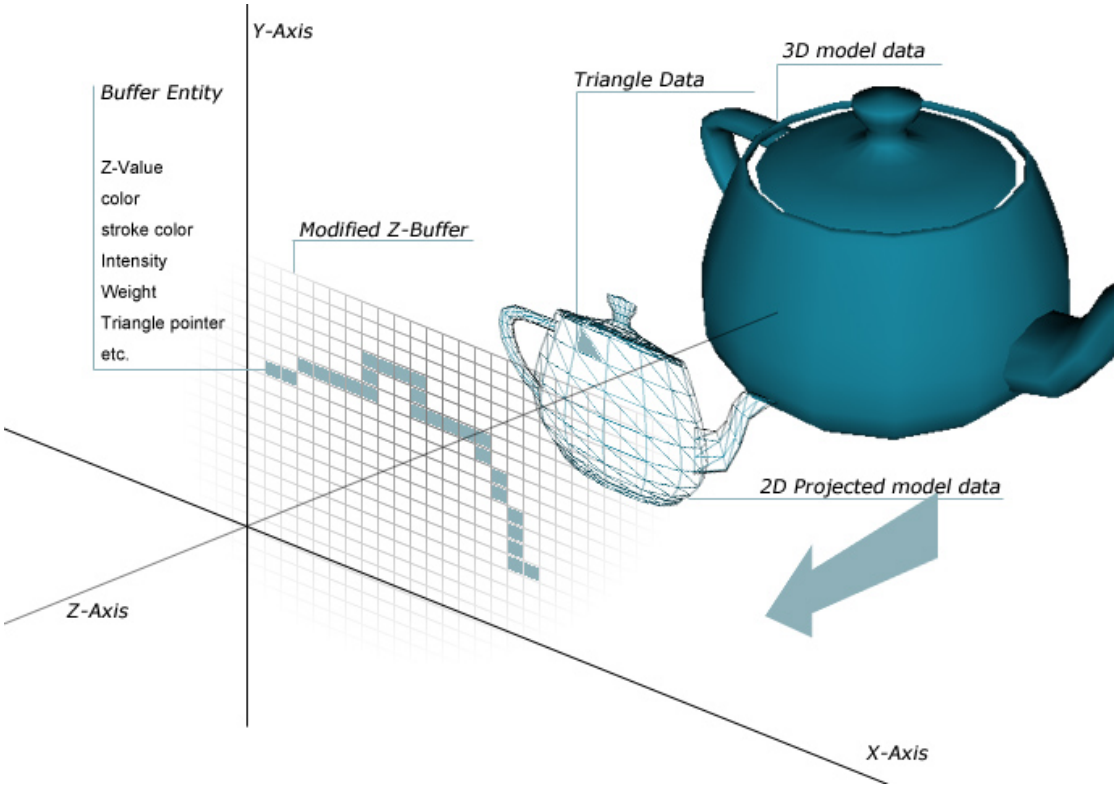
重新定義 Z-buffer 上的儲存資料結構來達到我們需要的結果。

原始的 3Dmodel 中的 Index 資料也必須重新整理。我們必須規劃出以 Triangle 為單位的資料結構，來儲存 Triangle-Triangle 之間的關聯性，以及每一塊 Triangle 各自的資訊，如線段資訊等。

3.2 系統架構

為了可以處理畫布上每一條筆觸的細節，我們建構出一塊由 Buffer 組成的虛擬畫布。此一畫布除了傳統的 pixel – Z information 之外，還必須能夠參考到原始模型的資訊，以及平面化後的 Shape 資訊，以便於計算出合適的筆觸資料。

另外，由於製作 Animation 的需求，模型的原始格式除了 Vertices、Indices 之外，還必須包含 Group，Bone，Frame 等資訊，故我們選擇 Direct3D 的.X file 作為模型資訊儲存的架構。

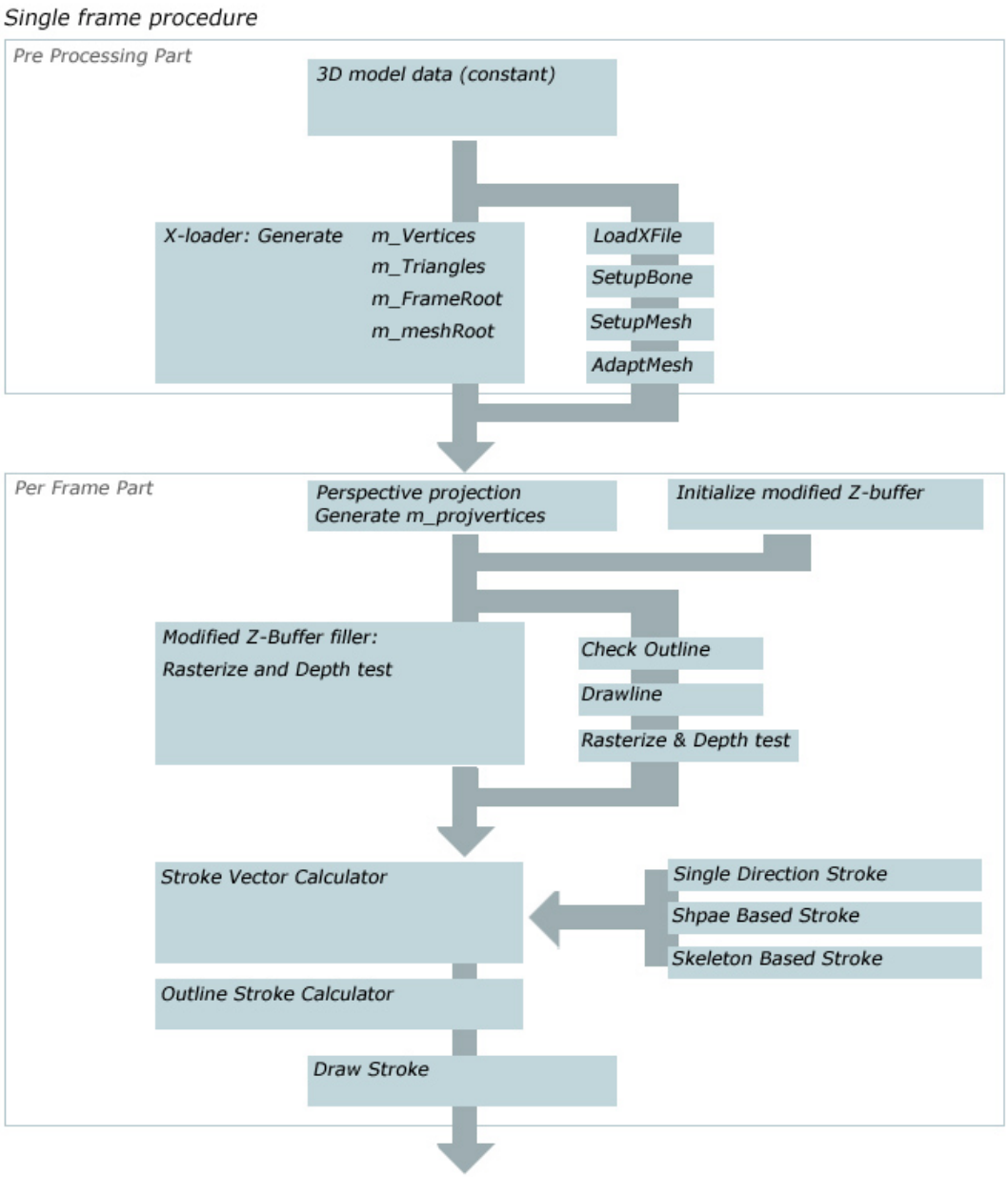


[圖十三] System architecture. 列舉幾個系統中重要元件的關係。

如圖十三所示，此為 NPR-System 的基本架構。3D model data 即為原始的模型資料；Modified Z-Buffer 即代表”虛擬畫布”。每一個 Buffer 上的像素必須儲存一個 Triangle data 的連結，以方便讓 Modified Z-Buffer 上的每一塊像素可以便利的參考 3D 原始資料，以及套用過所有的轉置矩陣(模型座標矩陣、世界座標矩陣、投影矩陣)後的 2D Projected model data。

其次，為了整理出模型 Triangle-Triangle 之間的相互關係，以及便利找尋模型上的輪廓線，我們必須重新定義原始模型中三角形的 Index 資料，重新儲存為 Triangle data。

3.3 系統運作流程



［圖十四］系統流程。包含前置處理部份以及單次影格處理程序。

本 NPR-System 概括來看，可以分為三個主要的部分。

(A)3D model loader – Pre Processing part

我們將由 X-file 讀入的資料，重新拆解並整理成我們系統需要的資料結構。模型與骨架的階層關係、模型內部的表面關聯性都在此作前置處理。

(B)Modified Z-Buffer: Rasterizing & Depth test – Per Frame part

1. 判別線段資訊。填入需要繪製的線段
2. 初始化 modified Z-buffer，並將 3D-model 的浮點資訊光柵化並填入 Z-buffer。
3. 深度測試，消除隱藏線。

(C)Stroke Calculator & Stroke Drawer

1. 選擇不同的筆觸生成方式，生成筆觸的走向。
2. 給定外框線段的權重。
3. 繪出筆觸效果。

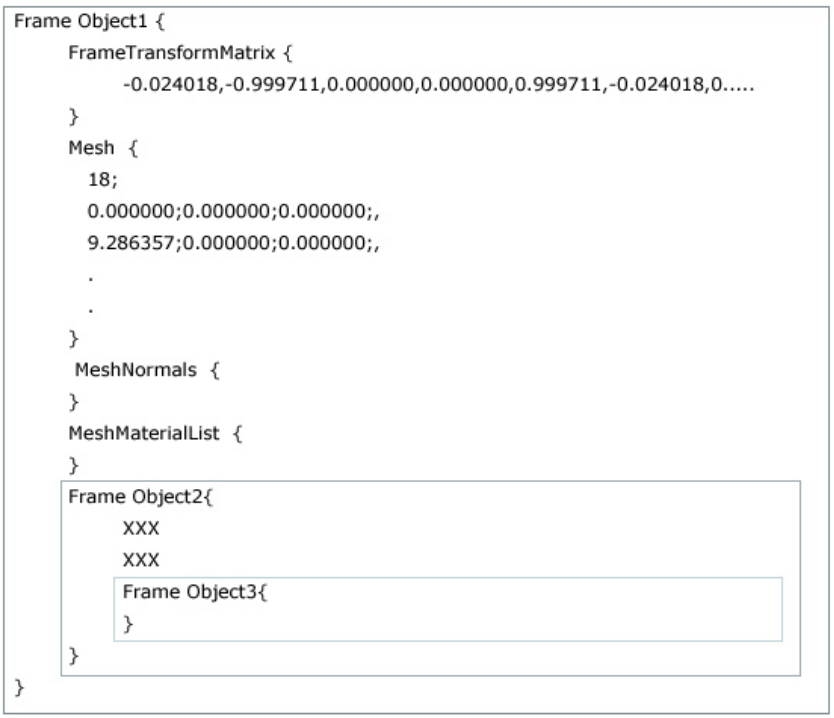
在接下來的章節，我們將依序介紹其實作方式。

第四章　資料的讀取及整理

在本系統中，早期的實作中本來採用.obj file format 作為 3D model 的資料格式。然而，為了能夠更進一步控制 Animation 以及骨架等參數，我們選擇使用 Direct3D 的 X-file 作為資料的儲存格式。另外，由於著名建模軟體如 Maya、3DsMax 皆有支援 X-file 的檔案格式，更便利於日後的相關開發。

4.1 X-loader

X-file 的資料格式，其優點為具有階層特性的物件關係。每一物件包含各自的 Group Name、Vertex、Index、Normal、UV、Bone 以及相關轉置矩陣。如圖十五所示。



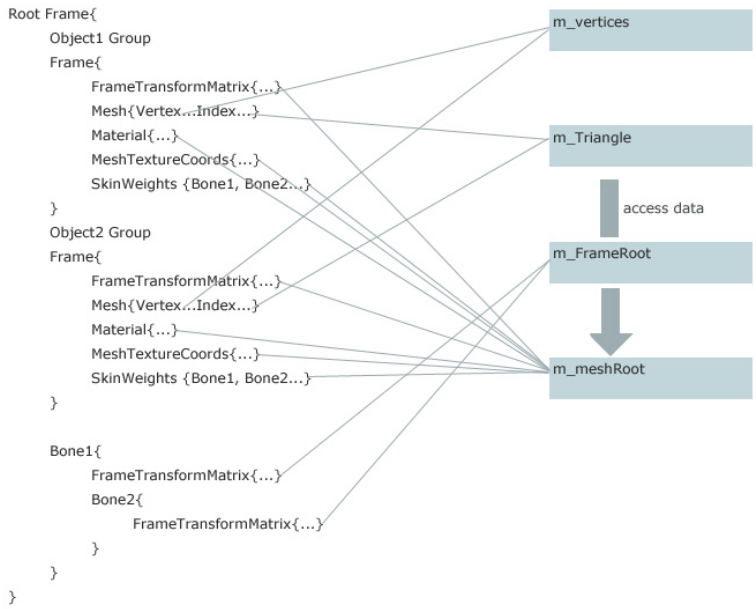
[圖十五] X-file 檔案結構物件的階層關係:Object1->Object2->Object3

4.1 X-file Adapter

由於 X-file 中的頂點資訊為”non-unique”的特性，若沒有經過整理與剔除多餘的頂點的步驟而直接使用，將會大量消耗不必要的系統運算資源。

因此，為了可以讓資料更適於自己的系統使用，我們在前置處裡的部分亦增加一個 Data Adapter，用來將 X-file 的檔案資料轉換成此 NPR-System 使用的資料結構。

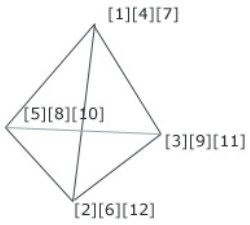
在 NPR-System 裡，我們將場景中的所有 Vertex 全部匯入整理成一份 Vertex List，而將群組關係及 Triangle 特性交由 Triangle List 管理。包含 Index 資訊，群組資訊，骨架資訊等。因此必須將 Input 的 X-file 重新拆解、整理。



[圖十六] Adapter 資料匯入示意圖。

(A)Vertices Data：

由各 Object 之 Vertex 資料取得。由於 Maya / 3DsMax 轉換的 X-file 其頂點資訊皆不具有唯一性，也就是說一個簡單的正四面體，卻會出現 3X4=12 個頂點資訊。因此，必須先做過一次資料篩選以及建立 Replace Table 更改其 Index 參照資訊之後，才能將具有唯一性的頂點資訊填入 Vertices List。



[圖十七]不具唯一性的 Vertex List - 出現 12 個頂點

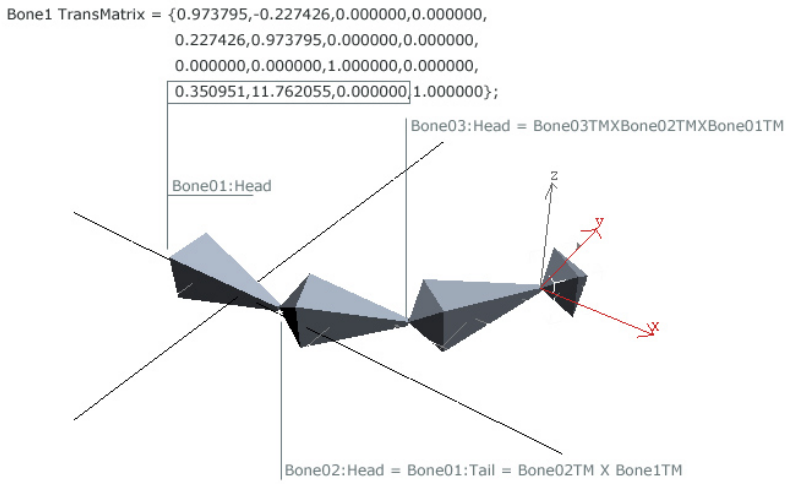
(B)Triangle Data：

根據上面的 Replace Table 以及物件間的 Offset Table 來更新資料資後，輸入到 Index List。

(C)FrameRoot：

儲存 Bone 的階層關係以及 Bone 的轉置矩陣、頭尾資訊等。

為了接下來能讓 Stroke 便於計算 Bone 的影響方向，我們從 Bone 的轉置矩陣以及其階層關係中，計算其對應的頭尾資訊。



[圖十八] Bone 的頭尾與其轉置矩陣的關係。

如圖十八所示，Bone01:Head = Bone01:TransMatrix(TM)中的 m41,m42,m43 項；

在計算 Bone02 時，Bone02:Head = Bone02:TM X Bone02:CombineMatrix。

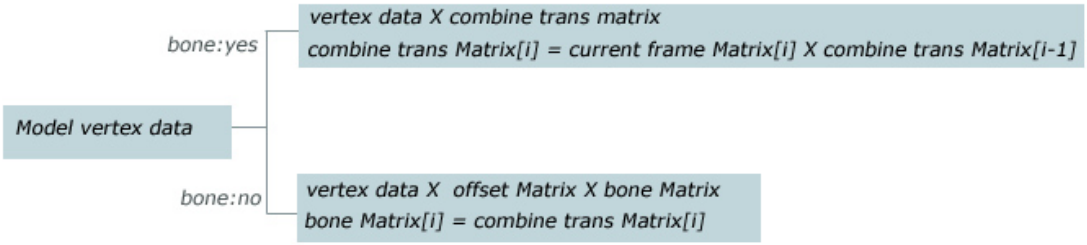
其中 CombineMatrix 則等於先前 Bone 的轉置矩陣的組合。

(D)MeshRoot：

儲存許多模型的相關資訊如 Material、FrameTransMatrix、Mesh Name、UV 等。

等待之後讓 Triangle List 由 Pointer 連結過來取得關聯的資訊。

此外，當讀取的 X-file 僅有基本的模型資料時，此 X-file Adapter 也負責轉換此種類型的模型資料，模型中 Bone 的有無，將決定 Vertex 所需參照的 Matrix。



[圖十九] X-file 中有 Bone 與否，將決定 Vertex List 所需參照的 Matrix。

第五章　虛擬畫布: Modified Z-buffer

我們使用自定的格式來實作一個 Z-buffer，即先前所提的虛擬畫布。使所有的 Stroke 計算以及生成均可以利用此畫布達到需求的效果。

5.1 Modified Z-buffer

```
class mZbuff {
public:
        mZentry buf[HEIGHT][WIDTH];

        int getSize();
        void clear();

        int pointNum;

        mZbuff();
        ~mZbuff();
};
```

```
class mZentry {
public:
        D3DXVECTOR3     vector;
        D3DXVECTOR3     strokeVector[3];

        DWORD           color;
        DWORD           strokeColor[3];

        float           intensity;

        TRIANGLE        *tPtr;
        LPSTR           meshName;

        bool            exist;
        int             type;
        float           Z;
        float           weight;

        mZentry();
        ~mZentry();
};
```

[圖二十] 兩個 modified Z-buffer 的主要資料結構。

如圖二十所示，右側的表為 Buffer Entity，代表每一個螢幕上出現的像素。

其中除了最基本的 Z value，也包含了關於此 Pixel 的權重(weight)、經過 3D 場景運算後的 color、以及各種筆觸的走向(vector[i])及相對應的筆觸顏色(strokeColor[i])等。

其中的 tPtr 則是用來連結所屬的三角形。透過此一指標，我們可以輕易的取得物體原本在 3D-model 所保留的資訊。

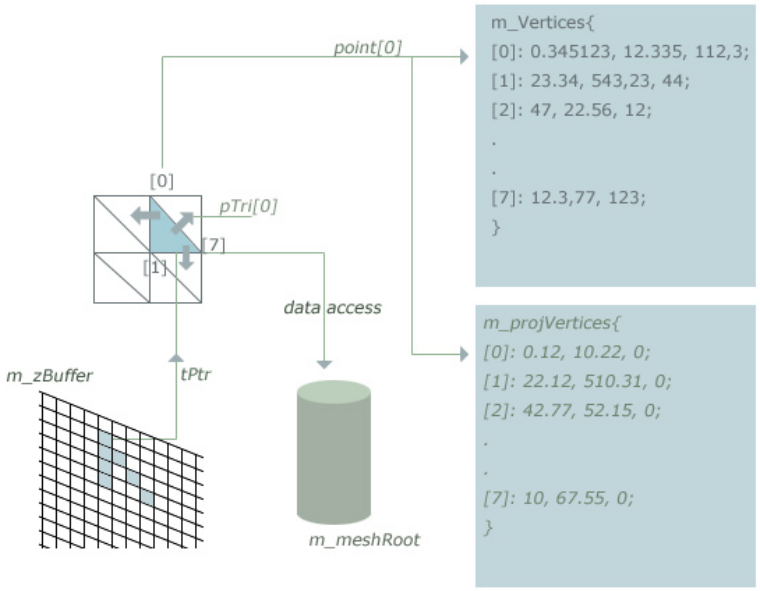
如圖二十一所示，TRIANGLE 的資料結構中除了基本的 Index 資訊以外，還有用來連結與周邊三角形之連結關係的 pTri[3]；儲存三角形三邊線段是否存在的 exist[4]；Triangle 所屬的 meshName；Triangle 所屬的 Material pointer 等。

```
class TRIANGLE
{
public:
	DWORD point[3]; // indices
	D3DXVECTOR3 normal;
	D3DXVECTOR3 color; //for debug

	D3DXMATERIAL *pMaterials; //直接連結位於mesh裡面的material資料
	TRIANGLE *pTri[3]; //[0]:0-1 [1]:0-2 [2]:1-2
	bool exist[4]; // [0]:0-1 [1]:0-2 [2]:1-2 [3]:the triangle

	LPSTR meshName;
	D3DXVECTOR3 sVector;

	TRIANGLE(){
		normal = D3DXVECTOR3(0,0,0);
		color = D3DXVECTOR3(0,0,0);
		for(int i=0; i<3; i++)
		{
			pTri[i] = NULL;
			exist[i] = true;
		}
		exist[3] = true;
		pMaterials = NULL;
		meshName = NULL;
	};
};
```

[圖二十一] TRIANGLE 資料結構

圖二十二說明了在系統中，Triangle List、Vertex、Buffer、MeshRoot 之間的關係。pTri[3]在 X-file Adapter 中，會將三角形與其週邊的三角形的關聯性儲存下來。如此可以讓之後的 Edge 判斷有可以參考的資訊。

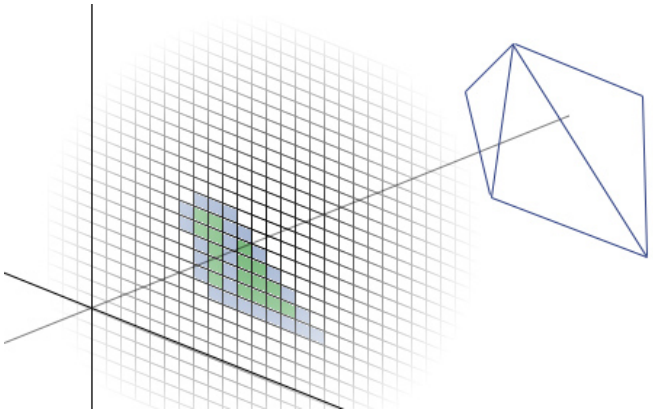
而 Point[3]的 Index 資訊則分別指到 m_Vertices 以及 m_projVertices 兩個頂點資料列。一個代表 3D-model 的原始頂點資料；另一個則是做過相關的模型座標，平面投影等矩陣運算後的頂點資訊。此外，每個 TRIANGLE 可以根據其所屬的 meshName 取得相對應的模型骨架資訊、以及 Material 資訊等。



[圖二十二] TRIANGLE-Buffer-Vertex-MeshRoot 之間的關係。

5.2 Rasterization

Rasterization 即是將模型上的 Vertex 浮點座標轉換成螢幕上的整數座標的一個動作。傳統的 Rasterization 僅將 Triangle 作填色的動作，我們的 NPR-System 為了稍後的筆觸運算，在 Rasterization 的同時，亦將 3D-model 的相關連接資訊傳入 Modified Z-buffer Entity 之中。



[圖二十三]Rasterization 示意圖。

如先前章節所言，物體的輪廓線可以分為三類：

(A)Silhouettes

(B)Boundaries

(C)Discontinuities

對於物體上的每一個 Triangle，我們檢測下列情形：

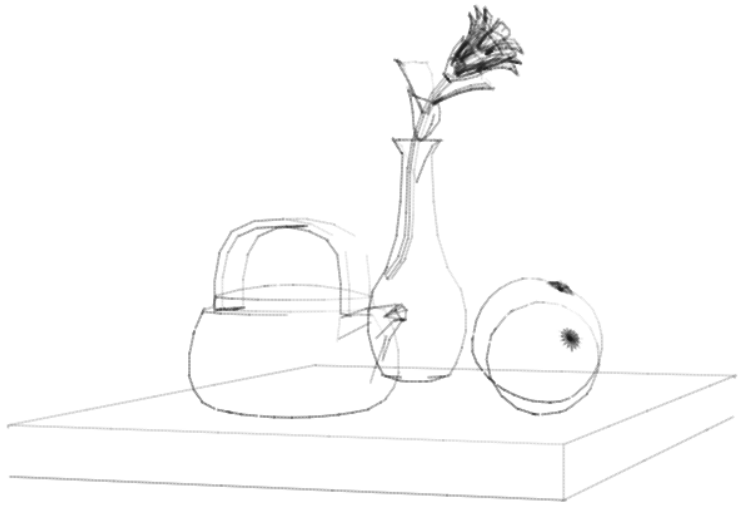
(A)其三角形相鄰的三個三角形，各自與視線的夾角。若夾角大於 90 度，則判定此兩個三角形中間所夾的線段為 Silhouettes。

(B)其三角形法向量與其相鄰的三個三角形各自的法向量之夾角。若夾角大於判別角，則判定此兩個三角形中間所夾的線段為 Boundaries。

(C)其三角形中，若有一邊沒有與任何三角形相連，則此邊判定為 Discontinuities。

(D)其三角形之法向量與視線的夾角。若夾角大於 90 度，則不計算該三角形之 Outline。(Cull Back Face)

然而，若單作上述的條件判別，所產生的圖形卻會發生錯誤。



[圖二十四] 綜合三種線段產生出來的 Output。

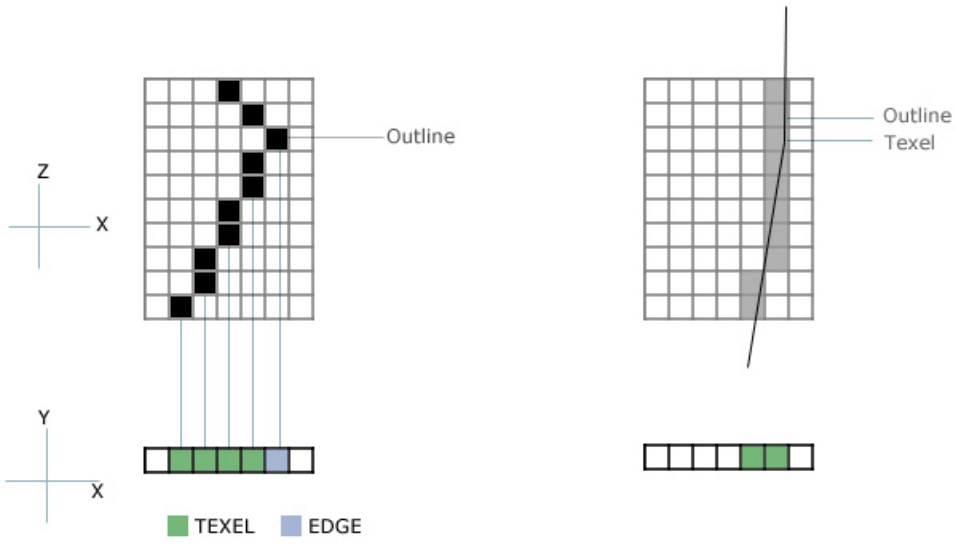
如圖二十四所示，花朵的莖部分與水果的下緣等「應該被前物遮蔽的線條」會出現在 output 之中。

藉由上述三項線段結合所產生的輪廓線，會發生從被遮蔽的物體後方透出來的情形，因此需要將這些「應該被前物遮蔽的線條」從線段中剔除，稱為 Hidden-Line Elimination。

為了消除這些線段，我們採用較為普遍的 Z-Depth Test 實作。對於場景中所有物件的 Triangle，利用 Rasterization，我們依序將其填入虛擬畫布(Modified Z-buffer)中，同時在每一個 Pixel 填入時，藉由比對 Z-buffer 中每一個填色物件的深度(Z value)，來決定最後此 Buffer 將歸於哪一個物件。

然而此一方法在實作時，對於我們的系統卻出現一個問題。NPR-System 中，我們將 modified Z-Buffer 單元分為三類：Empty、Texel、Edge。Empty 代表沒有被任何物件填色的 Pixel；Texel 代表被場景中的物件填色，且非該物件 Outline 的 Pixel；Edge 則代表場景中的物件填色、且為該物件的 Outline。

傳統 Z-Buffer 在使用時，考慮到的乃單純物件 pixel 的深度，並不需要考慮內圖色塊(Texel)與邊緣線條(Edge)的分別。因此，當物件為”高複雜度、圓弧狀表面”時，便會出現問題。



[圖二十五] 左：正常的物件作 Depth Test 所得結果；右：平滑表面所出現的問題。

當物件的平滑表面變成輪廓線時，由於過於平滑，用來判斷是否為 Outline 的 3D-model 邊緣會被自己的內圖色塊遮蔽(Z-Value 判斷後，texel 會遮蔽 Outline)。導致部份的 Outline 消失。

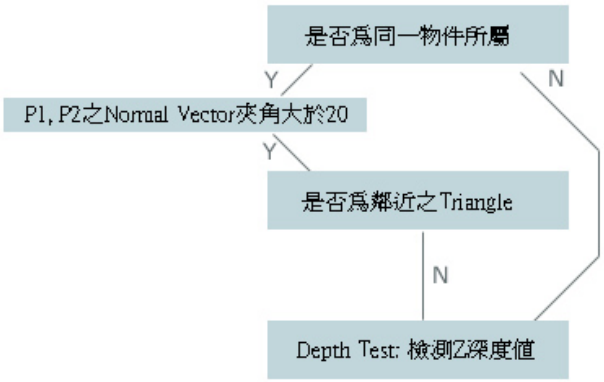
為了解決這個問題。我們改進了 Depth-Test 的檢測方法。基本檢測流程如下：

Rasterization:

```
For Each Triangle in Scene
    Rasterization
        for Each Pixel
            Adapt Depth Test
        end
    end of Rasterization
end
```

Adapt Depth Test:

令 P1 為目前儲存於 Z-Buffer 上的點、P2 為準備與 P1 作 Depth test 的點。

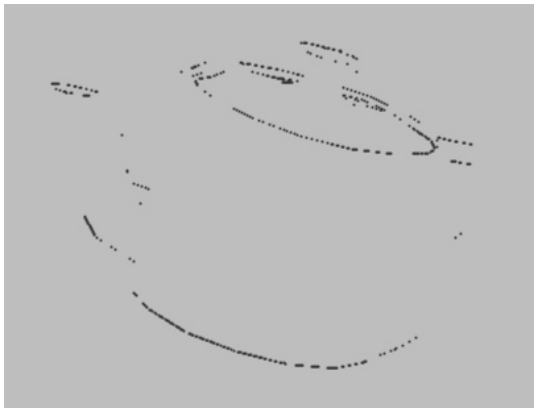


[圖二十六] Adapt Depth Test 流程圖。

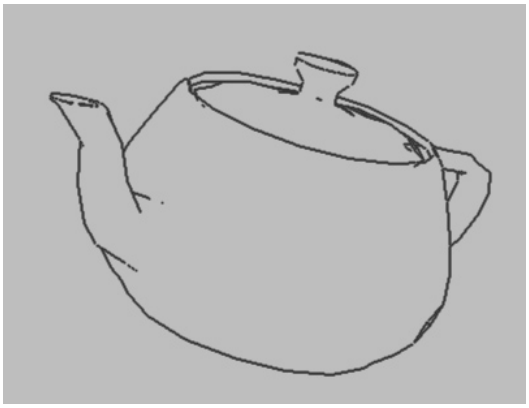
(A)若 P1, P2 分別屬於不同的物件，則直接作深度測試。

(B)反之，則判斷 P1, P2 點上的 NormalVector(由 3Dmodel 取得)之夾角，若超過一預設之門檻(20 度)則繼續判定條件(C)

(C)P1, P2 若彼此為非相鄰之 Triangle，則作深度測試。

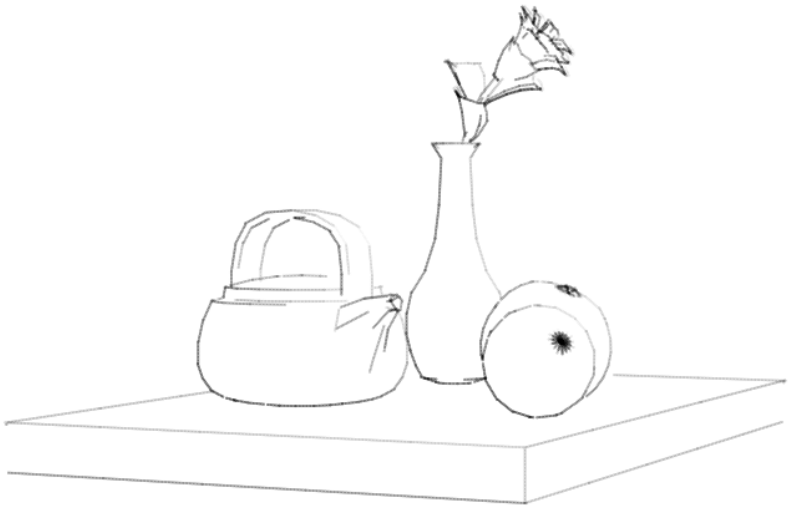


[圖二十七]套用傳統的 Z-Depth 的結果。Outline 被刪除的情形相當嚴重。



[圖二十八] 改進 Depth-Test 後的結果。Outline 正常顯示。

由圖二十七、二十八可以看見執行 Adapt Depth 後的結果。原本的深度測試法，當模型精細度越高時，產生的 Outline 遮蔽問題會越嚴重，改進深度測試的結果，便可避免大部分的遮蔽情況，順利的完成 Hidden Line Elimination。



[圖二十九] 執行完 Hidden Line Elimination 後的結果。

至此為止，我們已經將虛擬畫布 – 即 modified Z buffer 所需要的相關資訊整理完成。Buffer 上每一個像素都具有自己的屬性以及資料。等著接下來的筆觸特性計算。

第六章　光與影的生成

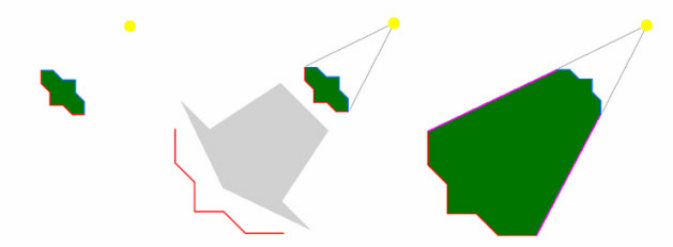
在 NPR 的影像生成中，我們並不需要過於精細的 pixel color，這是因為所有的 pixel color 僅僅擔任原始的畫面參考色，由這些參考色中經過 Layer、Blending 以及筆觸生成之後才生成最終的輸出影像。因此，在我們的 NPR 系統中僅採用簡單的 Flat Shading model 作為基本的 Shading 方法。

另外為了補足此簡單的 Shading method 所造成的不足，我們搭配 Volume Shadow 來生成必須的陰影區域。

6.1 Volume Shadow

Volume Shadow method 由一個 Shadow Mesh 與 Depth-Buffer 的 Depth Test 所構成。藉由 Direct3D 的 Shadow Mesh function 我們從原本的 Mesh 資料複製並重組為 Shadow Mesh，之後再與系統中的 modified Z-buffer 作深度測試。

執行完上述的工作，我們可以在 modified Z-buffer 中取得標記過的 shadow pixel。Shadow Mesh 由一個面光的 Front Face 以及背光的 Back Face 所構成。當我們將 Back Face 以光源圍投設點向後拉至無線遠識時，便可以得到一個延展的 Shadow Mesh。而被此 Shadow Mesh 所涵蓋住的物體，就是需要生成陰影的地方。如圖三十所示。



[圖三十] Volume Shadow 示意圖。

在我們的 NPR-system 中，由於 Render 所需要的 Z-buffer 以及基本成像方式的不同，Volume Shadow 必須重新改寫成適合我們系統的程式才能正確的顯示。故我們將上述 D3D 實作 Volume Shadow 的想法重新整理、Implement 成適合我們系統的程式，然後套用之。

第七章　虛擬畫筆: Stroke Calculator

為了能使生成的影像更接近於普通的繪畫，我們從先前的畫家繪畫中歸納出四種重要的特徵筆觸，這些筆觸各自有著不同的走向特性、不同的粗細特性等。

(A)Outline Stroke

(B)Single Direction Stroke

(C)2D shape-based Stroke

(D)3D model-based Stroke: skeleton

本章將依序討論這些筆觸的生成方式。

6.1 Outline Vector

由章節 5-2 中，我們可以取得物體的基本 Outline 線條。為了可以使這些線條看起來更具有線條的特性，我們進一步處理線條的粗細分布。紀錄在 modified Z-buffer 上的線段 pixel，每一個都具有原本的 2D 線段的 Vector 方向。

然而，這些方向是雙向性的。因為我們無法斷定每一個線條的走向。也就是說，在計算筆觸間的夾角時，向量 V(1,1)以及 V(-1,-1)會被視為相同的結果。在重整這些輪廓筆觸時，我們並不將所有的 pixel 重新分類為一組一組的線段，而是藉由尋找輪廓線中，具有指標意義的點作為影響其他線段特性分布的指標。我們認為輪廓上，轉折點以及線段匯聚處是影響輪廓粗細的一個指標。(底下定義為 Dominate Point, DP)

概念上，對於 Buffer 上所有的 outline pixel，我們先挑出所有線條的匯聚部分以

及轉角部分的 pixel，定義具有上述兩種特性的 pixel 為 DP。之後，其他的 outline 便可以尋找出離自己最近的 DP，依其彼此間的距離來決定筆觸的粗細。

欲找出所有的 DP，我們使用一個 3X3 的 filter 對整個 buffer 作遮罩運算。

對於在 buffer 上的每一個 outline pixel，進行以下運算：

(所有的 outline vector 皆為單位向量)

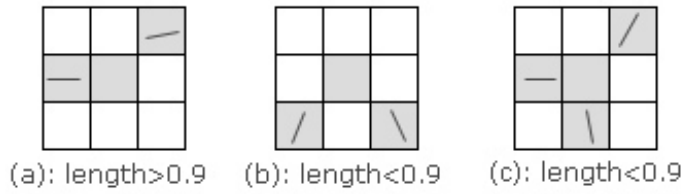
(a)計算除了遮罩本身的 pixel 以外，在 filter 內的 outline pixel 數量。

(b)加總除了遮罩本身的 pixel 以外，所有在 filter 內的 outline pixel 並平均之。

(c)計算出這個平均後的向量的長度。

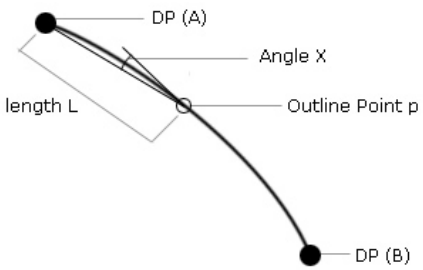
(d)設定取得 DP 的門檻值：0.9，若長度小於該門檻，則設為 DP。

如圖三十一所示。

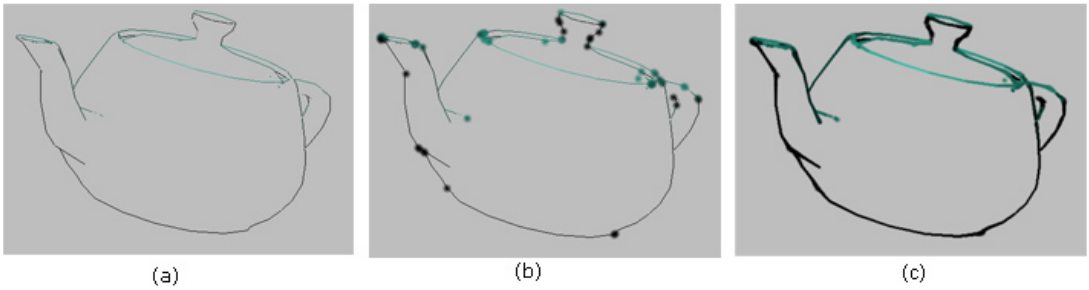


[圖三十一] (a):曲率低的線段部分，由於向量間的差異不大，故最終計算的長度也不會差原本的 1 太多。(b):轉折部分。兩個或多個差異大的 outline pixel，計算出小於 0.9 的長度值 (c):聚合部分。線段匯聚的越雜，其長度值遞減的越大。

找出所有的 DP 之後，便可以開始給定其他的 outline pixel 相對應的權重。如圖 29 所示，對於每一個 pixel，若可以在一定的範圍內(angleX)，找到距離最近的 DP，則此 outline pixel 的權重則可由該 DP 到 outline pixel 的距離決定。



[圖三十二] 如圖所示，DP(A)，以及 DP(B)皆於 angleX 內，Length L 小的 DP(A)影響了 outline point(P)的權重。



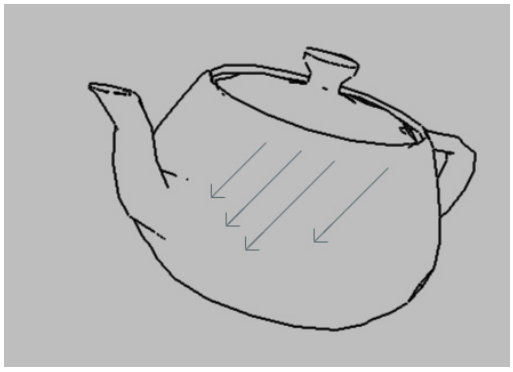
[圖三十三] (a)原始的 outline pixel 分布。 (b):DP 分布。 (c):分配完所有權重後所生成的輪廓線。

6.2 Single Direction Vector

單向性筆觸，其想法及實作方法皆簡單而單純。在油畫中常用來作為大面積繪畫前著底色的方式。

對於整塊虛擬畫布，我們灑上統一向量，方向為(0.7, 1)作為其 Vector Field。

之後，設定每一筆筆觸的最大長度，並以現有的 Outline 為中斷點，開始將畫布上的向量與顏色投入 LIC 中作疊加的運算。



[圖三十四]平塗法筆觸走向。

[圖三十五]套用平塗後的效果-填底色

6.3 2D Shape-based Vector

在繪圖的時候，有時我們需要額外的線條來強調物體本身 2D 的形狀。這種繪畫手法尤其在當物件本身具有不規律起伏的表面時。大致來說，此種繪線常會隨著物件本身的 Outline 起伏、生成。為了能夠模擬出具有此種特性的繪線。首先我們必須決定存在於虛擬畫布上的每一個 pixel，分別受到哪些 outline pixel 的影響。

在處理這個步驟上，我們首先將所有的 outline pixel 以各自所屬的物件分類之，以加速影響筆觸的搜尋。同時，這個分類的步驟也可以避免不同物件間，不必要的筆觸相互影響。

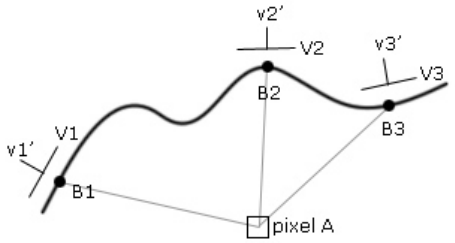
故，對於虛擬畫布上的每一個存在的 pixel，我們反覆執行以下的步驟：

(A): 搜尋所有影響到該點的 outline pixel。

(B): 給定一 threshold，判斷每一個 outline pixel 到該 pixel 的長度。若小於 threshold 則予以保留為 intensity pixel。

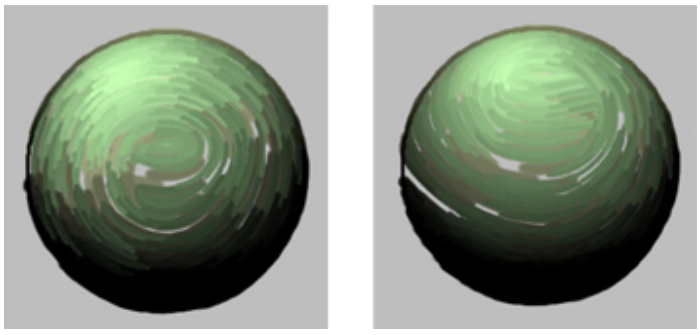
(C): 根據篩選出來的 intensity pixel，作最後加總平均的計算。

如先前章節所提，所有的 outline vector 都是雙向的。若直接將所有找出的 intensity pixel 直接加總，則我們無法統一所有筆觸的方向。因此，相較於加總所有 intensity pixel 上的方向向量，我們給定一個向量位於 intensity pixel 且垂直於該 pixel 上的方向向量，然後加總這個向量。如圖三十六所示。



[圖三十六] Pixel A 為運算中的向量；B1, B2, B3 為範圍內的 Intensity Pixel；相較於加總 V1，V2，V3，我們加總 V1’、V2’、V3’三個可以確定其方向的向量。因為 V’的方向可以根據作用中的 pixel 與其之間的夾角決定之。

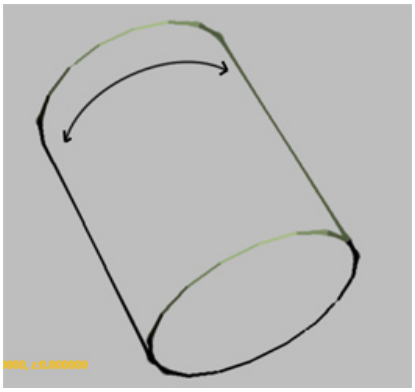
然而，若僅根據外圍輪廓來生成方向，這種風格的筆觸最終都會在影像的中心形成一個同心圓。我們認為這並非一個好的模擬結果。在一般的情況下，我們觀察到筆觸也會因為光源投影在物體上的位置而產生不同的筆觸變化。因此在這部分的筆觸走向生成中，我們額外加入光源對筆觸走向的影響。如圖三十七。



[圖三十七] 圖左：筆觸繞著中心形成同心圓；圖右：加入光源的影響後。擾亂原有的同心圓結構，改向亮光處繞行。

6.3 3D-Model based Vector: skeleton

當物件形體具有強烈的形狀特徵時，我們會依據模型本身的三維結構來給定其筆觸方向。



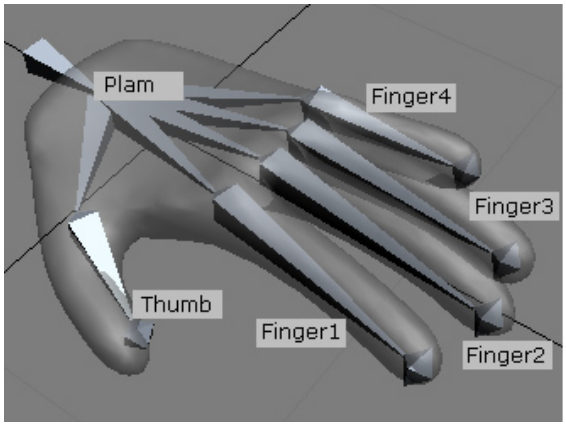
[圖三十八] 柱狀體，為建模軟體中的基本型體之一。通常我們使用如圖上所示的方向筆觸來加強觀者對物件型體的認知。

在傳統的繪畫中，物件的形體表現一直是重要的一環。而筆觸在這個環節所扮演的角色，往往是用來強調出物件的三維特性，讓觀者能由 2D 的繪畫在腦海中重建出物件原始的 3D 型態。即是說，筆觸往往能夠幫助觀者更正確的認知一幅畫中的物件。

為了模擬出這類型的筆觸，我們採用模型本身具有的骨架結構。從模型中抽取出的模型建構法，直至目前為止仍然沒有一個適用於所有模型的最佳演算法。在本篇論文中我們並不討論這個方向的題目，我們單純由目前市面上較著名的建模軟體，如 3DS-MAX、MAYA 等所提供的骨架結構抽取之。而在未來的工作中，我們將把目前較著名的骨架抽取法整合至本篇論文。

基本的骨架架構，由 Parent/Child 的階層結構組成。如圖三十九所示，palm 為 thumb 以及其餘四個 finger 的父階層。而每一個 Bone 會有其影響的範圍、權重以及轉置矩

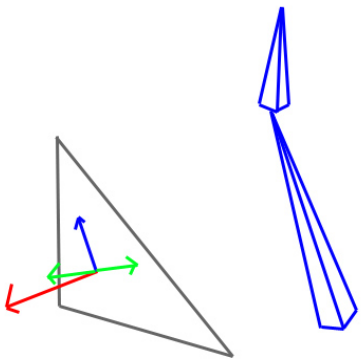
陣，骨架依據這些可取得的資訊來控制模型的動作。這是目前主流的模型操控方式。



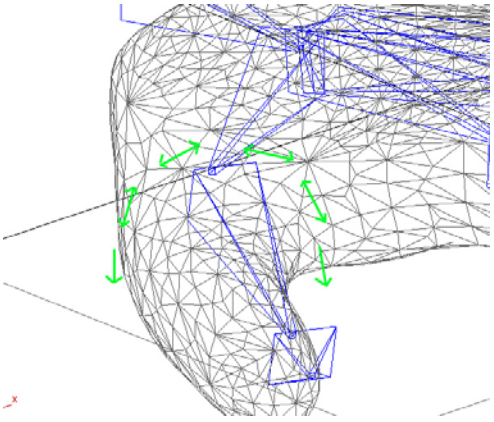
[圖三十九] 手掌的骨架階層構造。

我們從中取得骨架的相關資訊，即其個別的階層結構、範圍、權重以及轉置矩陣。以這些資訊來運算系統中的筆觸走向。

至此，根據每個 Bone 的影響範圍，我們可以讓每一個 Triangle 結構獲得一個 Bone 可供參考方向。外積此 Bone 方向向量與 Triangle 的法向量之後，可得到一個三維雙向性的筆觸方向，有了三維的方向向量之後，我們再對其作一次平面投影的矩陣運算後便可以獲得我們在 Buffer 上需要的二維方向向量。如圖四十、四十一所示。



[圖四十] 藍色箭頭代表骨架方向；紅色箭頭為 Triangle 法向量。綠色為兩者外積後的三維筆觸向量。



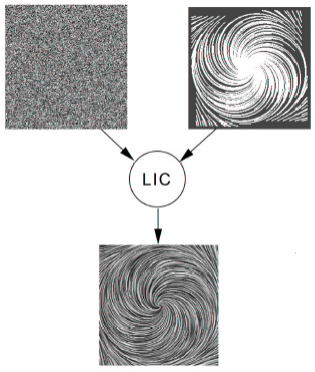
[圖四十一] 手掌上，Thumb 部分的三維筆觸分布。

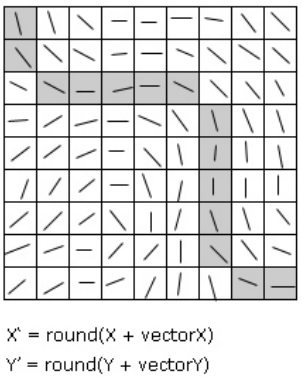
第八章　筆觸的渲染

有了前面章節所講述的筆觸向量生成法後，本章我們將討論如何將 Stroke Vector 與 Shading color 以 Line Integral Convolution (LIC)結合在一起。

8.1 Line Integral Convolution (LIC)

在本篇論文中，我們使用 Line Integral Convolution(LIC)來產生筆觸效果。簡單的 LIC 由一張 color/noise image 以及一張 Vector field 組成。對於 image 上的每一個 pixel，LIC 根據 vector field 追蹤一定長度的 pixel 並紀錄成一組組的 group，這一組組的 group 再經過顏色的混合運算，填色後，便能夠形成具有流紋效果的結果影像。如圖四十二所示。





[圖四十二] LIC 示意圖。Noise image 與 Vector field 輸入 LIC 的運算單元，生成具有流紋形的結果影像。

[圖四十三]Vector field 示意圖

我們利用這種流紋般的特性，來生成筆觸的效果。Noise image 以 shading 後，填入 modified Z-buffer 後的 color 來取代；vector field 則是以前一章節所提的幾種筆觸向量生成法則產生。

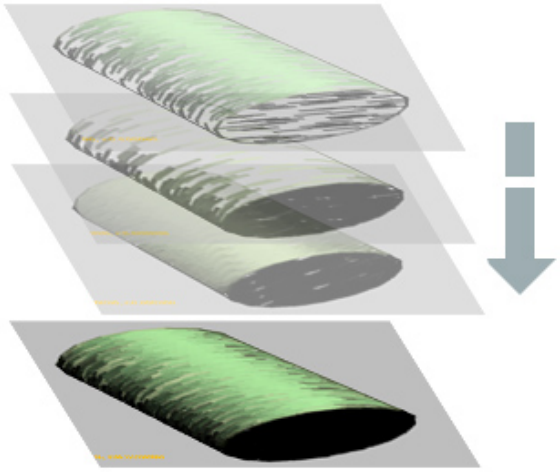
而 LIC 的運算單元中，對於 noise image 上的每一個點，我們根據其位置從 Vector field 找出對應的 Vector，累加，然後計算出下一個目標像素。如四十三所示。

而為了使整體筆觸看起來更平均，我們在整張 Vector Field 上灑出一定密度的 Seed，再利用這些 Seed 為起點，依照 Vector Field 的方向從兩端長出筆觸。

8.2 複合圖層

傳統畫家在繪畫的時候，往往不是只有一層的著色 。而是透過層層的顏色疊才完成最後的成品。這麼做一方面可以突顯畫作的層次感，從作畫的角度來看，也比較容易完成。大塊的顏色先打底，上基本色，由粗至細逐步完成畫作。

在我們所提出的系統亦沿用這樣的手法。首先我們使用單向、較粗的筆觸作打底的動作。其次以較粗的筆觸描繪暗部、最後再以較細、顏色差異較大的筆觸填出明亮的部分。



［圖四十四］筆觸分層機制。不同特性的筆觸分別負責不同區域的上色，最後再疊加起來，生成最後的影像。

第九章　筆觸風格

僅有筆觸的走向，並無法充分模擬出筆觸的特性。觀察畫家的筆觸可以發現，不同的運筆方式，也會產生剛度不同的筆觸，筆觸的粗細也會有變化。而筆鋒的特性不同，如乾、濕等亦會對筆觸的成像造成影響。在本章我們簡述我們所模擬的筆觸風格。

9.1 筆刷的選擇

目前市面上許多知名的繪圖軟體如 PhotoShop、Core Painter 等，可以透過更改不同的筆刷，造成不同的筆觸效果。這是目前比較主流的筆鋒模擬方式。在我們的系統中，亦能透過不同的筆刷選擇，來達到不同的模擬效果。

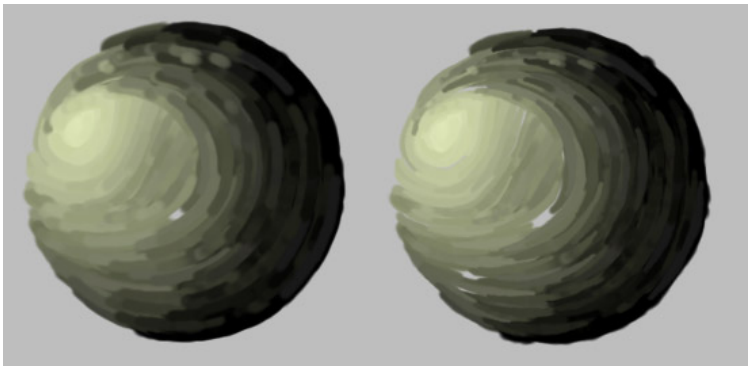
由於本篇研究並非著重於新的筆鋒模擬，故我們直接從 PhotoShop 取得其使用的筆刷並直接套用之。如圖四十五所示。

[圖四十五] 從 PhotoShop 所取出，各種不同的筆刷。

9.2 筆觸寬度模擬

繪畫時，隨著下筆的輕重變化，會造成整條筆觸的粗細分布不一的結果。相較於全部的筆觸粗細完全相同的結果，擁有粗細變化的結果終究是比較接近手繪的。然而由於

目前的研究成果仍在系統原形的開發上，故在我們的研究中，僅考慮最簡單的粗細模擬，即筆觸的兩端細、中間粗的基本形狀。



[圖四十六] 圖左：粗系相同的結果圖。圖右：加上粗細調整後的圖。

9.3 潦草筆觸的生成

從人的觀點來考量，在對一個物體上色的時候，並不會有完全吻合的筆觸，多少會有一些人為誤差使得筆觸稍稍偏離原本的位置。這些不完美使得繪畫的偏差值上升，也使繪畫更像是人所繪出的結果。

為了模擬這樣的誤差效果，我們讓每一筆筆觸在 LIC 生成時，其兩端的終點稍稍延伸出去，相較於使用亂數生成筆觸延伸的偏差值，我們以 Seed 的位置作為其延伸長度的偏差依據。以使讓筆觸可以在接下來的 Animation 部分連續。造成凌亂的筆觸效果。

第十章　動畫生成

本章節我們討論以一張張由我們的 NPR-System 所生成的影格連續播放產生的動畫。

10.1 筆觸連續性

簡單的動畫由連續的影像產生。動畫中的物件，通常一定有物件是會移動以及形變的。如同現今的卡通動畫一樣，分為前景及背景。由於在每張影格，前景都會有所變動，故前景的部分通常由簡單的色塊構成，以減少動畫師的負擔。而在我們的研究中，產生動畫最重要的課題在於如何使筆觸連續。

觀察許多的動畫作品我們可以發現，要使筆觸連續不外乎兩種主要的方法：

(a)以投影平面為基礎的筆觸連續。

視窗平面上的每一條筆觸，在 Frame 與 Frame 之間固定其生成位置。

(b)以模型為基礎的筆觸連續。

視窗平面上的每一條筆觸，會吸附在模型上，依照模型的形變而產生移動、增加或刪減的動作。

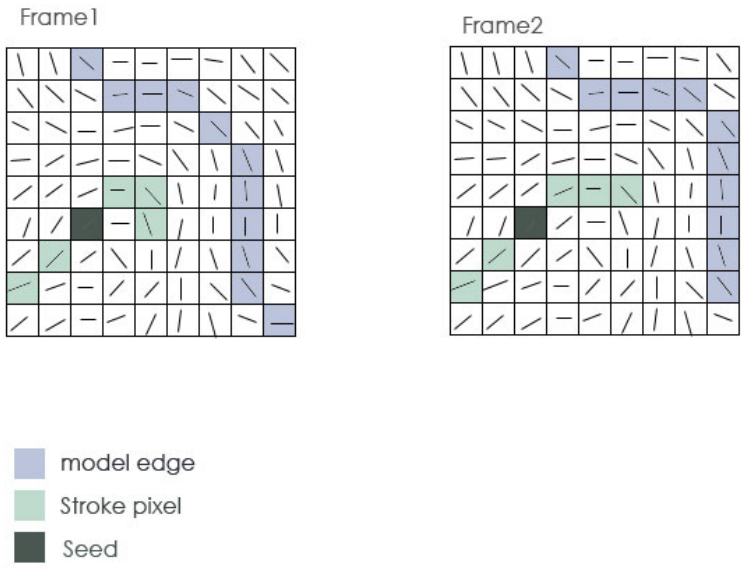
兩種方法都是為了讓眼睛所看到的連續影像不會過於”跳動”為前提，並沒有所謂絕對對錯的說法。

本篇研究我們採用較為簡單的(a)方法來實作筆觸連續性的問題。

如先前章節所提，筆觸從 Seed 的兩端長出。因此，Seed 可以決定一條筆觸的初始位置。為了讓筆觸能夠連續，我們使一開始灑出的 Seed 固定，之後在下一個 Frame 以

這些已經固定的 Seed 為優先生長對象，而因為物件的位移而超出筆觸生長範圍的 Seed 則刪除之。同理，新增加的區域則另外灑上新的 Seed 並紀錄其位置。

以此方法為原則，筆觸的生成會依據同一個 pixel 位置、套用不同的 Vector field 而生成具有”近似”卻不盡相同的筆觸。如圖四十七所示。



[圖四十七] 從 frame1 到 frame2，model edge 向右橫移了一個 pixel，但是筆觸的 Seed 位置不變。

第十一章　研究成果

本章節我們展示研究的成果。

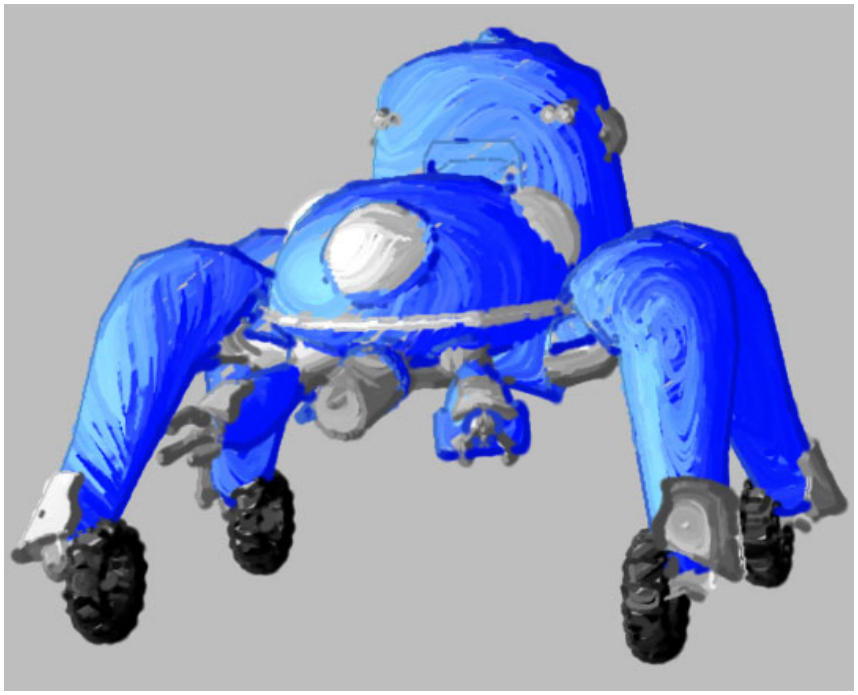
[圖四十八]左：正常的筆觸。中：使用較柔和、濕軟的筆觸。右：使用較乾燥的筆觸。

[圖四十九]正常筆觸生成：靜物。

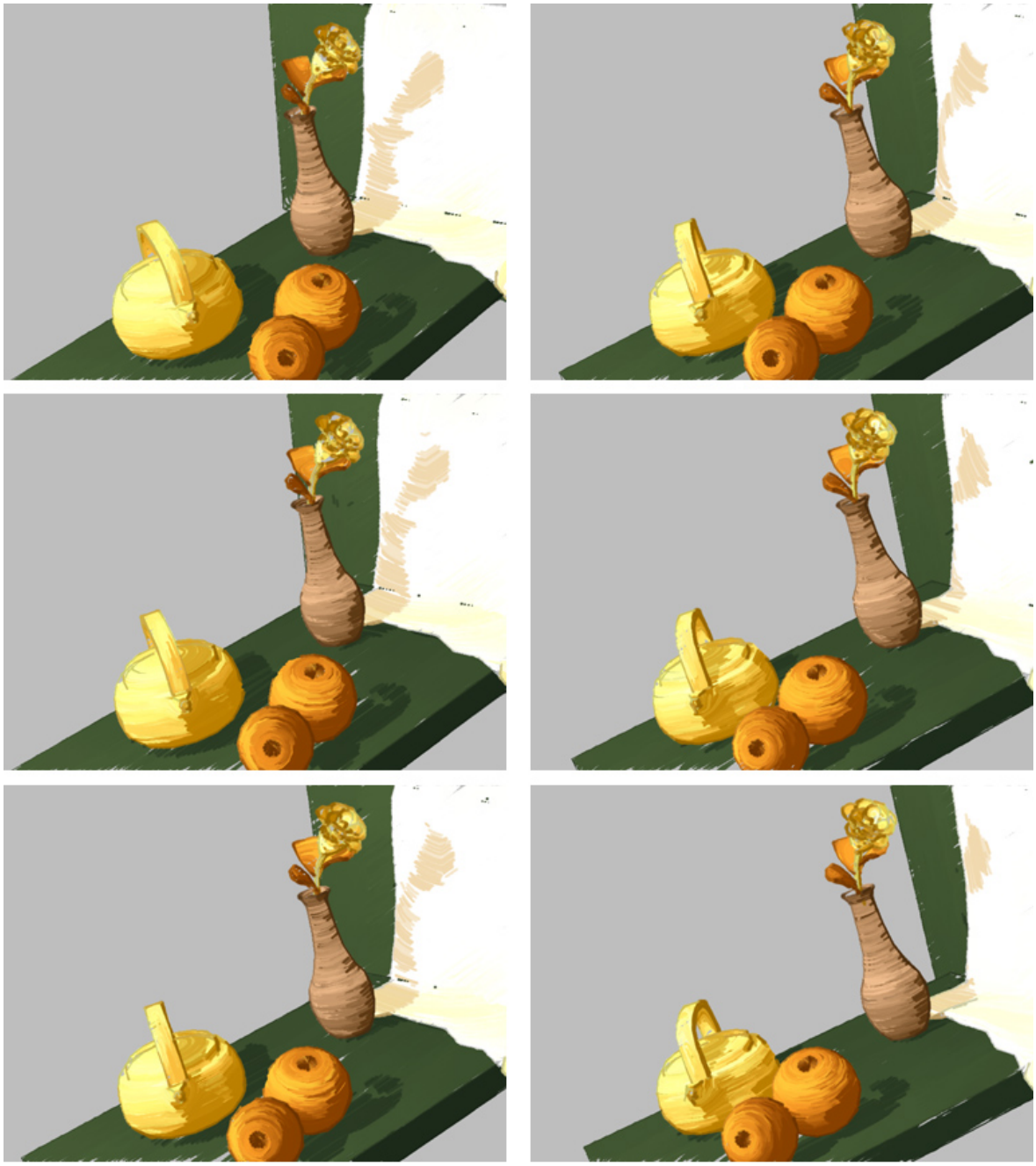
[圖五十]使用較凌亂的筆觸生成。

［圖五十一］高面數的模型

［圖五十二］高面數靜物場景



［圖五十三］Vehicle



［圖五十四］ 連續的靜物影格。

更多的圖像及影片展示，請見實驗室網站：http://163.22.21.164/web/index.htm

第十二章　結論以及未來工作

12.1 結論

在本篇研究中，我們提出一種新的 model-based NPR 生成方法來模擬連續的筆觸生成。藉由可任意取得模型本身 2D 以及 3D 的相關資訊，我們成功的藉由這些資訊模擬出各種筆觸特性的作品。而 LIC、Skeleton 以及複合圖層的整合，使我們能夠更近一步模擬畫家會畫的手法，進而生成生動的筆觸。

本篇研究中所使用的各種筆觸走向，各有其特點，根據需要描繪的目標物特性的不同，可以選擇出最適合描繪該目標的筆觸走向。舉例來說，當目標物具有明顯的形體特性時，可以使用 3D-skeleton 的筆觸來描繪之；當模型具有複雜曲率變化的表面時，則適合用 2D-edge 的筆觸走向來呈現。

複合圖層的使用，能夠更近一步模擬出畫家繪畫時常用的疊加手法，透過不同特性的筆觸疊加出具有層次感的畫作。另外，我們也採用了 Volume Shadow 的陰影生成方式，替我們的系統在光影的呈現方面更具真實性。

12.2 未來工作

本篇研究藉由Modified Z-buffer，將NPR與3D-model based的許多概念進行整合。目前只有針對筆觸的走向以及特性處理，其餘如顏料的特性變化等，都尚欠缺考慮，另外，仍有許多圖學的特性如Global Illumination等，可以加入系統中。

底下列出可在未來改進此系統的幾個方向：

(a): 模擬非固體的目標體以及各種特效的生成。

目前系統只有針對可以模型化的物體作處理,若要進一步模擬出無法模型化的物體如煙霧、流水、火光等，則需要額外的資料結構來生成。

(b): Skeleton 的自動生成。

目前雖然有許多的 Skeleton 抽取演算法已經開發出來，但仍沒有一個可以針對所有物體,Optimal 的演算法。我們希望日後 Skeleton Extraction 的演算法更加成熟後，能將其套用到目前的系統中，讓原本不具有 Skeleton 資訊的物件能夠藉由該演算法自動取得其本身的骨架資訊。

(c): 更多筆觸特性的開發。

我們相信仍有許多的筆觸繪畫特性是我們所沒有觀察到的。藉由目前已經開發出的系統平台，希望能在未來觀察出更多的筆觸，並且模擬之。

(d): 更精準的顏色生成。

目前系統採用的 Shading model 為 Flat Shading。雖然搭配 Volume Shadow 已有不錯的光影效果,但我們仍希望在未來能夠模擬出物件與物件之間光線折射所產生的細微顏色變化。故未來可考慮使用 Ray Tracing 或 Radiosity 等光源計算方式取代 Flat Shading。

(e): 改寫為 Plug-in。

目前主流的建模軟體如 MAYA 或 3DS-MAX 皆有提供 Plug-in 的編寫方法。我們希望能夠將此研究以 Plug-in 的方式嵌入建模軟體中，成為獨具一格的 Render 方法。讓更多的使用者接觸這項研究成果。

參考文獻

[1] Daeuk Kang, Donghwan Kim, Kyunghyun Yoon, "A Study on the Real-Time Toon Rendering for 3D Geometry Model," iv, p. 0391, Fifth International Conference on Information Visualisation (IV'01), 2001.

[2] Appel, FJ Rohlf and AJ Stein. The haloed line effect for hidden line elimination. ACM Computer Graphics (Proc. of SIGGRAPH '79), 13(2):151--157, August 1979.

[3]. Lake, A., Marshall, C., Harris M., and Blackstein, M. Stylized Rendering Techniques for Scalable Real-Time 3D Animation. In Proceedings of NPAR 2000: First international symposium on Non-photorealistic animation and rendering, Annecy, France, pp. 13-20, June 2000.

[4] P. Haeberli. Paint by numbers: Abstract image representation. SIGGRAPH Conf. Proc., pages 573--580, 1990.

[5] Michael A. Kowalski, Lee Markosian, J. D. Northrup, Lubomir D. Bourdev, Ronen Barzel, Loring Holden, John F. Hughes:
Art-based Rendering of Fur, Grass, and Trees. 433-438
Electronic Edition (ACM DL) BibTeX

[6] Oliver Deussen, Thomas Strothotte. Computer-Generated Pen-and-Ink Illustration of Trees. SIGGRAPH 94. p 91 - 100.

[7] R. Galimberti and U. Montanari. An algorithm for hidden line elimination. Communications of the ACM, 12(4):206--211, April 1969.

[8] Level Set Diaglrams of Polyhedral Objects. Francis Lazarus*, Anne Verroust.

[9] J.P. Hall, P.M. Painterly rendering using image salience. Eurographics UK Conference, 2002. Proceedings. The 20th. P 122- 128

[10] Thomas Luft, Oliver Deussen, "Non-Photorealistic Real-Time Rendering of Characteristic Faces," pg, pp. 339-347, Computer Graphics and Applications, 12th Pacific Conference on (PG'04), 2004.

[11] James Hays, Irfan Essa. Image and video based painterly animation. SIGGRAPH 2004. p 113 - 120.

[12] Han-Wei Shen, David L. Kao. A New Line Integral Convolution Algorithm for Visualizing Time-Varying Flow Fields. IEEE Transactions on Visualization and Computer Graphics, 1998. p 98-108.

[13] Adrian G. Bors. Introduction of the Radial Basis Function(RBF) Network.

[14]. Mark J.L.Orr. Introduction to Radial Basis Function Networks.. Centre for Cognitive Science. University of Edinburgh,Buccleuch Place,Edinburgh. 1996.

[15]. Eric B. Lum, Kwan-Liu Ma, "Non-Photorealistic Rendering Using Watercolor Inspired Textures and Illumination," pg, p. 0322, Ninth Pacific Conference on Computer Graphics and Applications (PG'01), 2001.