國立交通大學

電機資訊學院 資訊學程

碩 士 論 文

三維電腦遊戲引擎之中國水墨風格
人物白描技法模擬

The Simulation of Chinese Ink Style Figure
Painting Techniques in 3D Game Engine

研 究 生：余冠鳴

指導教授：施仁忠 教授

中 華 民 國 九 十 四 年 六 月

三維電腦遊戲引擎之中國水墨風格人物白描技法模擬

The Simulation of Chinese Ink Style Figure Painting Techniques in 3D Game Engine

研 究 生：余冠鳴　　Student：Kuan-Ming Yu

指導教授：施仁忠　　Advisor：Dr. Zen-Chung Shih

國 立 交 通 大 學

電機資訊學院 資訊學程

碩 士 論 文

A Thesis

Submitted to Degree Program of Electrical Engineering and Computer Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Computer Science

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

三維電腦遊戲引擎之中國水墨風格人物白描技法模擬

研究生：余冠鳴　　指導教授：施仁忠 教授

國立交通大學電機資訊學院 資訊學程（研究所） 碩士班

摘　要

無論中外，在早期繪畫的發展史上，都以人物畫爲主。中國古代人物畫表現的技法上大致可分成三類， 一爲白描畫法、二爲工筆重彩畫法、三爲寫意畫法（簡筆畫法）。然而白描畫法是人類最早，最簡潔的繪畫表現形式，中國古代稱爲「白畫」，也有人稱爲「線描」，於明嘉慶年間鄒德中等人將歷代畫家使用的筆法彙集十八種描法，將白描的技法做了分類。本論文之研究重點，爲將三維電腦遊戲引擎中繪圖機制，改以電腦自動找出模型輪廓及特徵，並以調整線條的粗細變化及渲染的筆觸方法，來繪製三維電腦遊戲中的三維人物模型，使人物模型的物體呈現白描技法的效果。在繪圖速度方面，我們提出了狀態機的繪圖方式，將傳統每次重畫全部場景的方式，拆成若干個狀態來繪圖，以提昇三維電腦遊戲引擎對輸出入裝置之反應速度，使畫面顯示與遊戲之操作得以更加順暢。

The Simulation of Chinese Ink Style Figure Painting Techniques in 3D Game Engine

Student：Kuan-Ming Yu Advisor：Dr. Zen-Chung Shih

Degree Program of Electrical Engineering Computer Science
National Chiao Tung University

ABSTRACT

No matter China and other countries, the early painting relies mainly on people drawing in the history of development of drawing. In Chinese ancient people drawing, it can be roughly divided into three kinds on technique displayed: a technique of drawing in brush style, color painting of the Chinese realistic drawing, and the comfortable technique of drawing. However, the drawing technique in brush style is that the people is the earliest and use the most succinct form to express. It is called ‘pure-draw’, also called ‘line-draw’. The De-Zhong Zhou et al. collected 18 kinds technique of drawing in the Ming Dynasty. It classifies the technique of the drawing in brush style. In this thesis, we modify mainly the drawing mechanism of three-dimensional computer game engine. It is automatically to find out silhouettes and characteristics of model with computer, and to present the result of Chinese ink drawing by adjust the thickness of lines in brush style. In speed of drawing aspect, we propose a state machine render mode. It is divided into several states rendering that is the tradition rendering method to render all scenes each time. The goal is to improve the reaction speed of three-dimensional computer game engine, and to smooth display and play.

致　　謝

首先我要感謝指導教授　施仁忠教授的教導與指引，才能完成此論文著作。我還要感謝電腦圖學實驗室的所有學長同學們的事實幫忙，讓我得以順利克服程式的障礙與觀念上的困惑。尤其是這幾年一同搭車的蔡玉寶學長給我很大的鼓勵，還有魏德樂學長的國畫素養提供我極大的幫助與及蔡侑庭學長在程式上的協助，使能完成此一系統架構。另外我要感謝我的兩位同事蔡耀廣先生與林偉煌先生在象棋模型的全力支援與建議，使我的實作得以順利進行。最後我要感謝我的家人，尤其煮點心妻子的支持與鼓勵，讓我得以專心於論文研究。非常感謝大家！

目錄

中文摘要……i

英文摘要……ii

致謝……iii

目錄……iv

表目錄……vi

圖目錄……vii

一、 序論……1

1.1 論文動機與目標……1

1.2 系統架構概觀與流程簡介……2

1.3 電腦遊戲引擎評估……9

1.4 章節概要……14

二、 相關研究……15

三、 繪圖演算法……18

3.1 水墨畫模組……18

3.1.1 毛筆模組與移動控制機制……18

3.1.2 水墨筆觸模擬機制……25

3.1.2.1 墨色變化……25
3.1.2.2 中鋒與側鋒模擬……28
3.2 三維模型資訊萃取……30
3.2.1 找出輪廓線與特徵線……31
3.2.2 去除隱蔽線……33
3.2.3 線段連結與控制點……34
3.3 白描技法模擬……37
四、 繪圖狀態機……39
4.1 動畫繪圖狀態機……39
4.2 靜態繪圖狀態機……42
4.3 效果分析……44
五、 電腦象棋……46
5.1 簡介……46
5.2 人工智慧演算法……47
六、 實作結果……51
七、 結論……57
參考文獻 ……59

表目錄

表 1 動態狀態 A 的真值表.....41
表 2 動態狀態 B 的真值表.....41
表 3 動態狀態 C 的真值表.....42
表 4 動態狀態 D 的真值表.....42
表 5 靜態狀態 A 的真值表.....43
表 6 靜態狀態 D 的真值表.....44
表 7 執行速率表列.....55
表 8 場景與模型的三角面與頂點數量.....56

圖目錄

圖 1 三維電腦遊戲引擎系統架構……4
圖 2 三維電腦遊戲引擎 NPR 模組……4
圖 3 NPR 繪圖處理步驟……5
圖 4 系統執行步驟……8
圖 5 Quake III 三維電腦遊戲引擎的軟體執行架構……11
圖 6 Half Life 三維電腦遊戲引擎的軟體執行架構……12
圖 7 Fly3d 三維電腦遊戲引擎的軟體架構……13
圖 8 毛筆中鋒與側鋒的模擬範例(a)中鋒，(b)側鋒……19
圖 9 毛筆與畫布接觸區域之定義……20
圖 10 以等角切割成數個區域……22
圖 11 毛筆的移動軌跡與旋轉……25
圖 12 墨色變化效果……26
圖 13 墨色在Y方向作線性灰階變化產生的效果……26
圖 14 不同的墨色擾亂程度產生的效果……27
圖 15 筆毛斷水效果……27
圖 16 毛筆壓力與接觸面積成正比……29
圖 17 側鋒接觸面由單邊提筆……29

圖 18　輪廓線之定義……31
圖 19　被選擇的棋子與場景的ID參考影像（爲加快執行速度）……34
圖 20　線段連結演算法……35
圖 21　不同參數 *t* 的 *Cardinal Spline*……36
圖 22　動畫繪圖狀態圖……41
圖 23　靜態繪圖狀態圖……43
圖 24　（a）模型邊緣外的筆觸被覆蓋，（b）保留模型邊緣外的筆觸……45
圖 25　遊戲樹範例……50
圖 26　PR Silhouette 場景……52
圖 27　柳葉描模擬（stroke_mode = 0）……52
圖 28　減筆描模擬（stroke_mode = 1）……53
圖 29　竹葉描模擬（stroke_mode = 2）……53
圖 30　琴弦描模擬（stroke_mode = 3）……54
圖 31　棗核描模擬（stroke_mode = 4）……54
圖 32　綜合技法模擬（stroke_mode = 5）……55

一、序論

1.1 論文動機與目標

中國繪畫，以人物畫發展爲最早，最初是以描繪當時的生活寫實爲主，人物的線條描繪由原本石刻形式演變爲筆法簡勁流暢的畫意，以及後來參照書法家的筆法創出一種完全以墨線完成畫面的「一筆法」[1]，以墨踪描繪的表現技法，爲後世所稱的「白描」。而白描人物是僅以墨線來勾勒成畫，而不加色彩，完全以墨線本身具有的繪畫效果來表現。人物畫像是以線來寫型，運用線條的曲直、剛柔、粗細、簡繁、輕重來表達畫意。於明嘉慶年間鄒德中等人將歷代畫家使用的筆法彙集成十八種描法[1,2]。這十八種分別是：「高古游絲描」、「琴弦描」、「鐵線描」、「行雲流水描」、「馬蝗描」、「釘頭鼠尾描」、「混描」、「橛頭釘描」、「曹衣描」、「折蘆描」、「柳葉描」、「竹葉描」、「枯柴描」、「蚯蚓描」、「戰筆水紋描」、「減筆描」、「橄欖描」與「棗核描」等，而上述十八種技法依線條之變化可分類爲無粗細變化（鐵絲描類）、有粗細變化（蘭葉描類）及快速減化筆線（減筆描類）等三種。

然而在目前電腦繪圖技術的發展上，欲呈現中國水墨風格的研究仍處於初期的發展階段，尤其於需求快速的繪圖速度應用上仍屬困難的，最普遍的方式仍是以傳統貼圖或

告示板的方式呈現，例如市面上號稱有水墨效果的 ”軒轅劍-蒼之濤” 電腦遊戲，亦是以傳統的貼圖技巧來呈現。有見於此本論文的目的是企圖以毛筆模組模擬運行的機制，讓電腦自動繪製出具水墨渲染與筆觸的效果，實作上我們呈現以線條變化爲分類的三種白描技法，分別是無粗細變化的行琴弦描、有粗細變化的柳葉描與棗核描、快速減化筆線的減筆描與竹葉描，最後是前述的綜合描法。雖然我們的實作效果無法與畫家手繪比擬，但是率先將此水墨的繪圖風格應用於三維電腦遊戲引擎的先例。在繪圖速度方面，我們提出了狀態機的繪圖方式，將傳統每次重畫全部場景的方式，拆成若干個狀態來繪圖，以提昇三維電腦遊戲引擎對輸出入裝置之反應速度，使畫面顯示與遊戲之操作得以更加順暢，另外我們也將象棋的人工智慧演算法實作進來，使之成爲一個實質的三維立體象棋程式。

1.2 系統架構概觀與流程簡介

最近幾年NPR（Non-Photorealistic Rendering）的研究是項新的研究領域，而NPR的目標是呈現畫家的各種藝術臨摹技法效果，而不是繪製真實的世界影像，若要如實的模擬畫家的各種藝術臨摹技法效果，在目前的NPR發展上仍有一段長遠的路要走。我們可以很容易從先前的NPR研究中發現，幾乎大部分的研究都著重於西畫的模擬上，像水彩繪[5]、影線畫[8, 19]、鉛筆畫[30, 31]與及剛筆墨畫[27, 28, 29]等皆是，相對於少數的中國水墨畫NPR的研究，實在令人感到遺憾。典型的中國水墨畫是結合簡單的線條來傳遞深遠的情懷意境，不是以豐富的色彩來表現圖案，而要模擬中國水墨畫的效果不全然是瑣碎的事，已有自由的毛筆線條、畫布上的表面皺摺、墨水的擴散與沉澱等相關的研究[13, 18, 35]。

本論文之研究重點，爲將三維電腦遊戲引擎中繪圖機制，改以電腦自動找出模型輪廓線及特徵線，是採用暴力法來完成此項任務的，並以調整線條的粗細變化及渲染的筆觸方法，來繪製三維電腦遊戲中的三維人物模型，使人物模型的物體呈現白描技法的效果，此項我們參考 Weng [35]所提的水墨模組模擬機制來完成的。而在電腦繪圖的技巧與步驟上我們參考 Northup [20,22] 等人提出的方法來實踐。在電腦繪圖的速度方面，我們提出狀態機的繪圖機制[25]，將傳統每次重畫全部場景的方式，拆成：全部繪製（並儲存影像）、先繪靜態部份（並儲存影像與深度值）再繪動畫物件、回存靜態部份深度值再繪動畫物件與將儲存影像直接輸出至螢幕等四個不同繪圖狀態，用以提昇三維電腦遊戲引擎對輸出入裝置之反應速度，使畫面顯示與遊戲之操作得以更加順暢。圖 1 顯示三維電腦遊戲引擎系統架構，圖 2 顯示三維電腦遊戲引擎 NPR 模組，圖 3 我們顯示包含繪圖狀態機在內的 NPR 繪圖處理步驟，而圖 4 則顯示系統執行步驟，其後我們說明主要的功能項目。

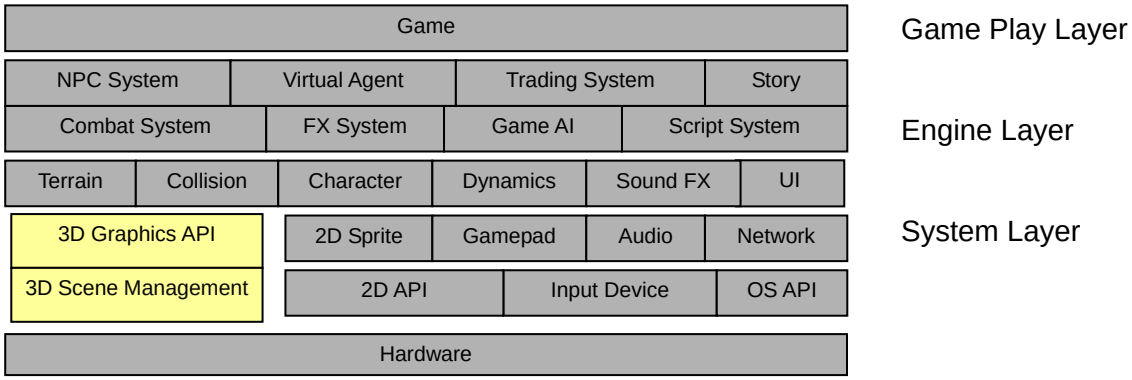


圖 1 三維電腦遊戲引擎系統架構

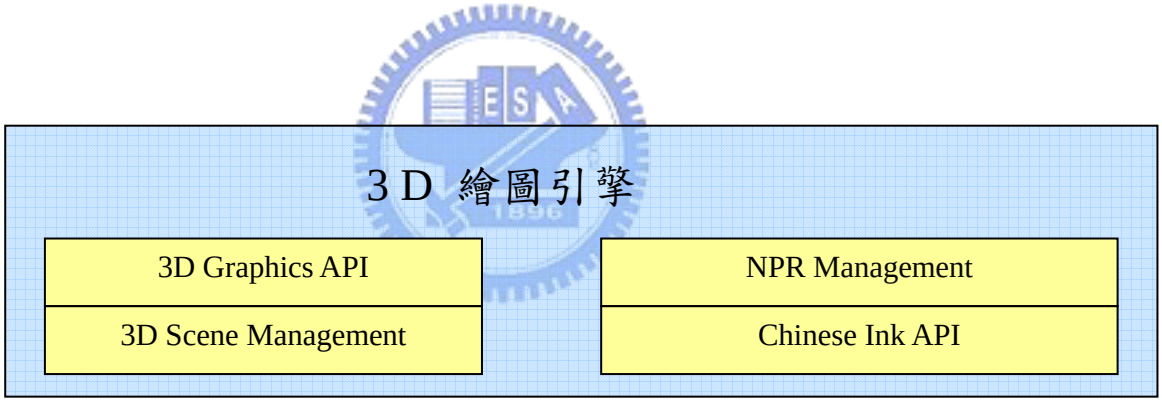


圖 2 三維電腦遊戲引擎NPR模組

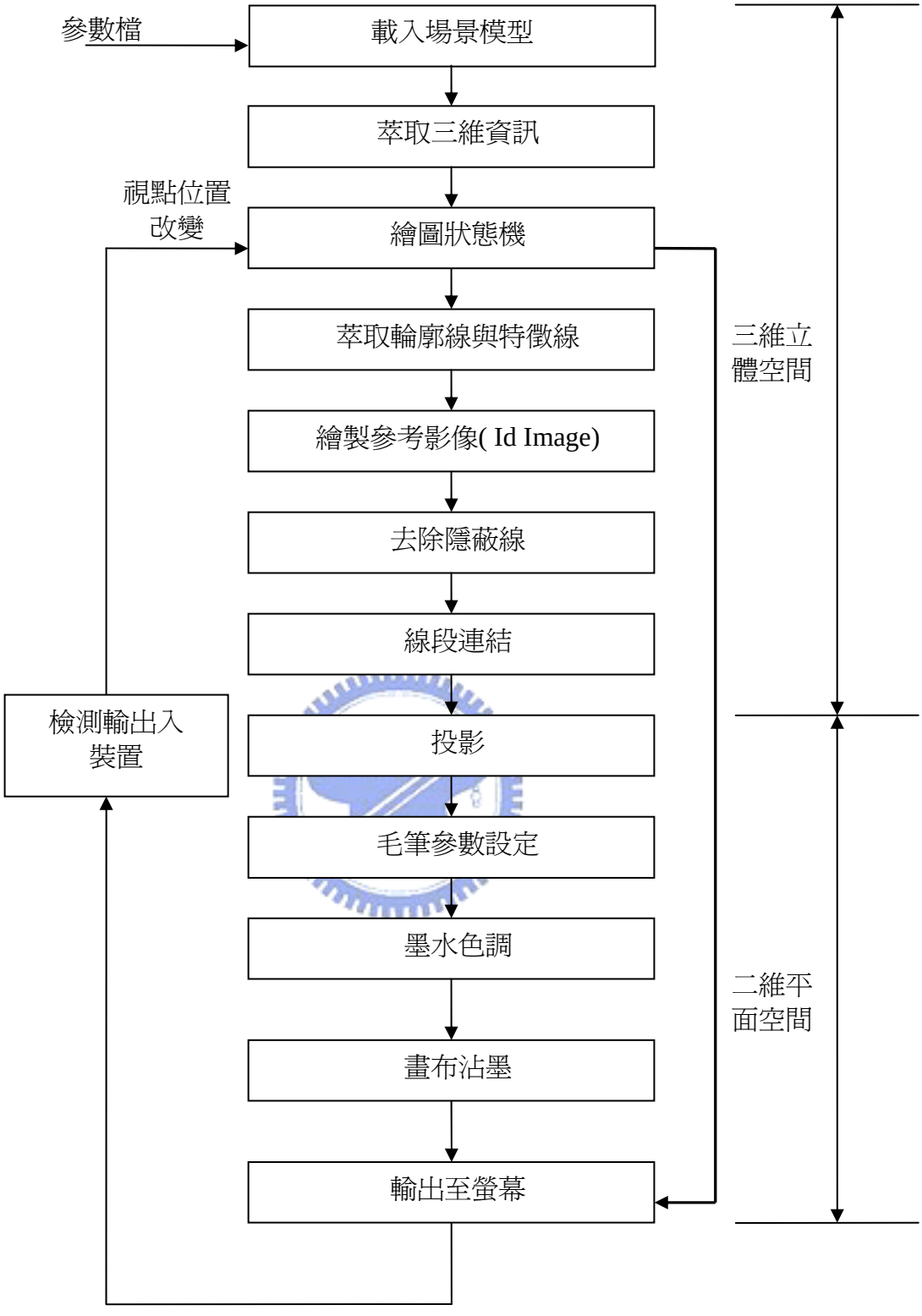


圖 3　NPR繪圖處理步驟

1. 載入場景模型(BSP、3DS 與 FAO)：首先，三維電腦引擎根據 .fly 參數設定檔案，載入三維場景模型（.BSP檔）、靜態物件模型（.3DS檔：由3D Studio Max匯出）以及動畫物件模型（.FAO檔：由Fly3d工具程式轉換）。

2. 萃取三維資訊（3D Information Extraction）： 此步驟是先計算由3D Studio Max匯出的場景模型與物件模型的法向量（Normal）、邊線（Edges）與邊界方塊（Bounding Boxes），再以一維陣列的資料結構方式，存於三維電腦引擎動態聯結模組中（.DLL），系統執行時再依索引值來讀取對應的三角面相關資訊。

3. 繪圖狀態機（Render State Machine）：此步驟會根據目前繪圖所處的狀態，及狀態變化的相關條件，經邏輯運算後選擇新的狀態：全部繪製（並儲存影像）；或先繪靜態部份（並儲存影像與深度值）再繪動畫物件；或回存靜態部份深度值再繪動畫物件；或將儲存影像直接輸出至螢幕。

4. 萃取輪廓線與特徵線（Feature Lines Extraction）：此步驟電腦會自動的找出三維場景中物件模型的輪廓線及特徵線，而模型的輪廓線與特徵線我們會根據不同的視點位置，在每一個視框（Frame）以暴力法方式重新計算獲得，這個步驟是在物件三維的空間中進行的。

5. 繪製ID參考影像(ID Reference Image)： 此步驟，我們仍然以OpenGL的繪圖機制來繪製，不同的是我們會給予每一個可見的線段，指定一個唯一且大於128 的顏色編號，供下面步驟處理參考，此繪製出的影像就稱作ID參考影像[20]。

6. 去除隱蔽線（Hidden Lines Removal）： 此步驟，我們會從視框緩衝區（Frame Buffer）中直接讀取ID參考影像，再以二維平面方式偵測及判斷先前繪製的三維模型的可見輪廓線與特徵線是否爲其他物件所遮蔽，重新標示是否爲可見的小線段。

7. 線段連結（Segment Link）：此步驟，是根據上個步驟偵測出的可見小線段，將之連成較長的線段，供下面步驟產生較平滑的曲線。

8. 投影（Projection）：此步驟，將可見的較長線段投影到二維平面，再根據較長線段的起點與終點，計算產生***Cardinal Spline*** 的控制點，最後再依毛筆模組的模擬機制模擬於曲線上。

9. 毛筆參數設定（Brush Parameter Setting）：此步驟，是載入墨水的參數設定值，並起始毛筆模組物件，並設定毛筆模組參數及應用物理基礎的毛筆模擬機制[35] ，來繪製中國水墨風格的效果。

10. 墨水色調（Ink Tone）：此步驟，是根據毛筆參數設定值，起始墨水的色調以呈現我們要的結果。

11. 畫布沾墨（Ink Diffusion on Rice Paper）：此步驟，於畫布上模擬出墨水的運行效果[13]。

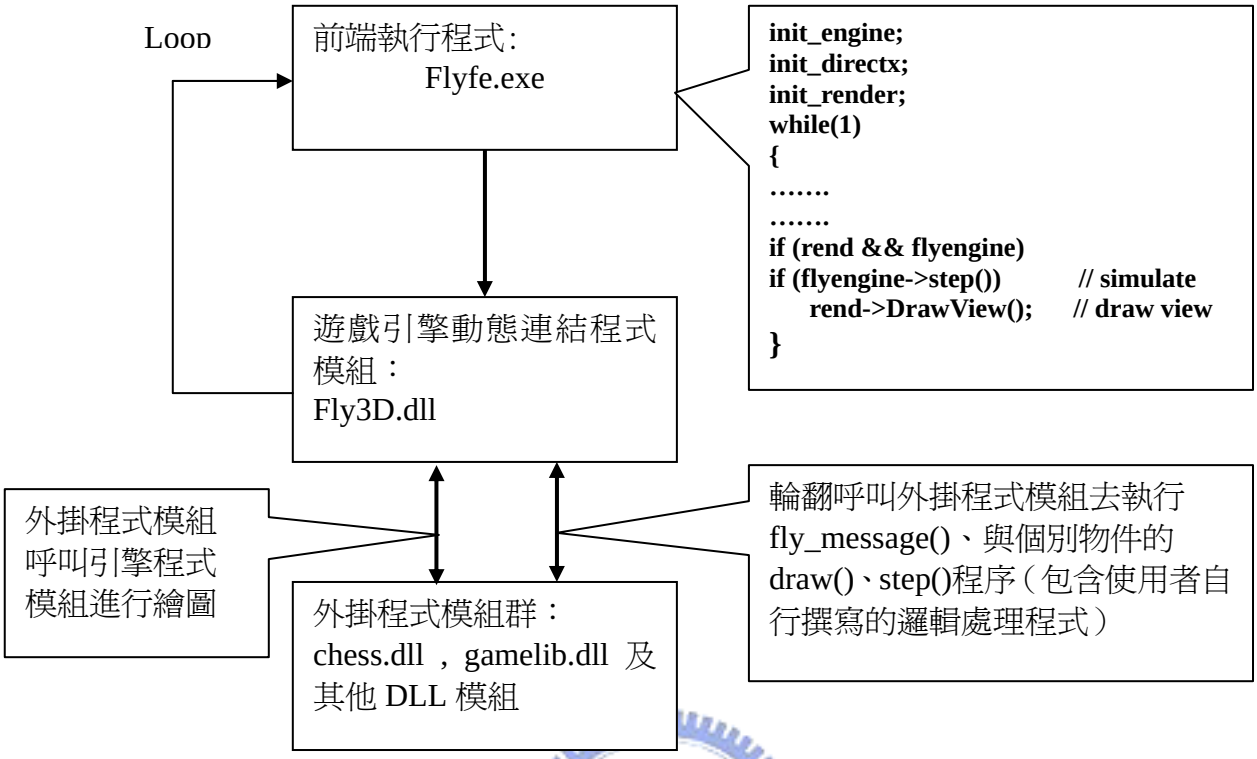


圖 4 系統執行步驟

1. 前端執行程式（Front-end）： 此前端執行程式主要功用是，起始電腦遊戲引擎程式模組、繪圖程式模組與 DirectX 程式模組，最後進入無窮回圈，輪翻的呼叫外掛程式模組去執行 fly_message()、更新視點位置與個別物件的 draw()、step()程序。

2. 遊戲引擎動態連結程式模組（Engine）： 此模組爲整個系統的核心，實際負責場景模型與物件模型的資訊儲存、OpenGL 繪圖程序、偵測輸出入裝置、播放聲響及其他用到的程序。

3. 外掛程式模組群（Plug-in）： 此外掛模組程式群，原電腦遊戲引擎提供的外掛程式模組有 gamelib.dll、lights.dll、viewport.dll、weapon.dll 與 menu.dll 等可供利用，而

使用者可另外針對不同的須求自行撰寫外掛程式模組加以擴充，例如：我們自行所撰寫的人機介面控制程式與參考使用的象棋人工智慧處理程式等。

1.3 電腦遊戲引擎評估

爲使我們的研究得以順利進行，我們參酌評估了幾種在電腦遊戲市場上較普遍有名，以 Quake 爲系列的第一人稱三維電腦遊戲引擎，以及我們所使用的三維電腦遊戲引擎 Fly3D。而 Quake 目前已發展至 QuakeIII 但都以 C 程式語言所開發的，而 Half-Life 則改以 C++物件導向程式語言開發，由 Quake 系列修改而，可說是如出一轍，然而爲什麼會受到電腦遊戲市場的青睞，其主要原因是，該系列的三維電腦遊戲引擎免費提供了玩家參與遊戲內容製作的主要編輯工具以及網路上龐大工具群，例如：關卡（level）場景編輯器（Valve Hammer Editor）、貼圖編輯器（Wally Texture Editor）、天空貼圖編輯器（Sky Editor）及其他小的編輯工具。

以 Quake 系列的三維電腦遊戲引擎在場景模型上的要求，需是一個封閉的模型空間，在此密閉的模型空間中，再以天空貼圖的方式來呈現戶外的天空景色，另外在場景中的各種實體如：靠近時門自動開關、踩到不同物體發出不同的聲音、靠近某個物體出現或發生某種事件等等，都需要有一個對應負責的實體程式，亦就是說玩家可以自行編輯設計遊戲內容，在場景中加上自己喜歡或創新的實體，以呈現自己獨特的遊戲風貌，但玩家需就這些新增的實體，自己撰寫對應的實體程式，重新編譯到遊戲引擎中，做爲自豪或考驗他人的作品。由此我們可以了解 Quake 系列之所以受歡迎的原因，我想是此一成功的商業模式所締造成的，但可惜的是商業軟體往往並不提供、也不允許我們去修改他們的核心部份，畢竟是爲了自己的商業利益著想。在我們評估三維電腦遊戲引擎

時，已有一個以 Quake 爲主的 NPR-Quake 遊戲研究[39]，他們的方法主要是攔截前端程式（Client）周期性送給後端程式（Engine）的指令，而蒙騙後端程式去執行自行轉寫的 NPR 繪圖程式，以呈現 NPR 的風格，然而其效果並不好，閃爍的非常嚴重，可說只是三維 NPR 電腦遊戲的一個例子。

評估後我們選擇 Fly3D V1.02 三維電腦遊戲引擎[38]作爲我們論文研究的實作系統，其主要的優缺點如下：

- 一個免費的三維電腦遊戲引擎教學範例，提供引擎所有的原始碼，V1.02 版模型的儲存結構（Triangle）很符合 NPR 的計算需求。而新的版本 V2.0 提供了更多更複雜的資料結構（Triangle、Quadrangle 及 Large Polygon），較不適合我們的實作。

- 以 3D Studio Max 作爲場景編輯器的三維電腦遊戲引擎，比前述 Quake 系列的三維電腦遊戲引擎提供的編輯工具，更具設計自由度，功能更強。

- 玩家自行增加與設計的外掛程式模組，相較於 Quake 系列的實體程式，需撰寫較底層的程式如：物件的座標轉換計算、物件與物件之間的邏輯處理、人工智慧程式與輸出入裝置的控制偵測等，都需要玩家自己撰寫，這對初學者而言較爲困難，是爲缺點。

接下來我們分別顯示 Quake III、Half-Life 與 Fly3D 三維電腦遊戲引擎的軟體架構：

- Quake III 三維電腦遊戲引擎

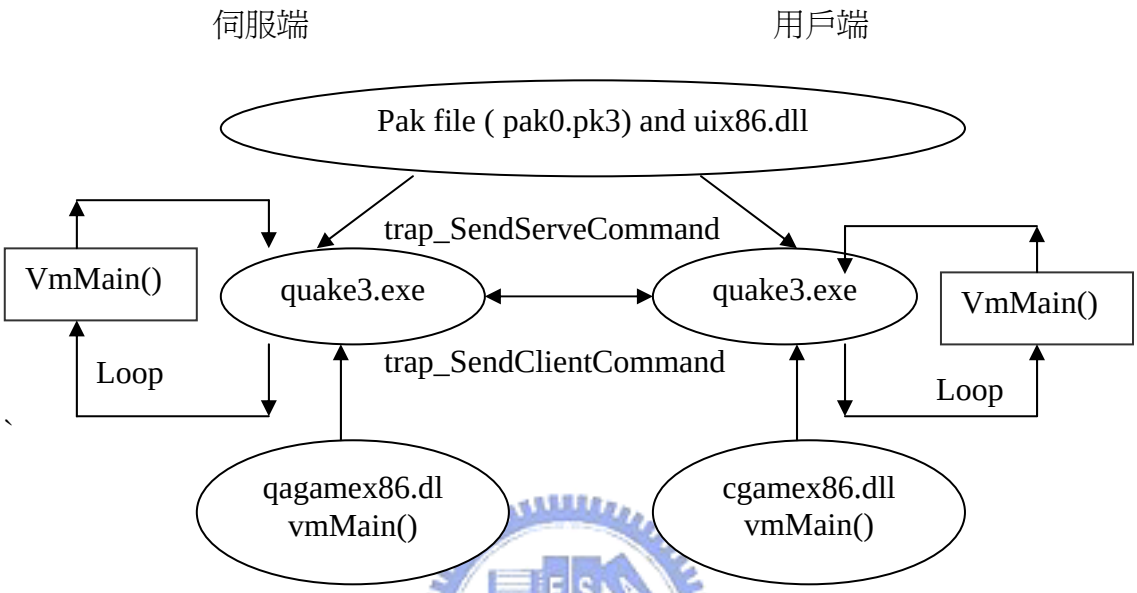


圖 5 Quake III 三維電腦遊戲引擎的軟體執行架構

1. qagamex86.dll ：爲遊戲引擎邏輯的核心，功能包含邏輯處理、繪圖、機械人人工智慧判斷與遊戲狀態紀錄等。.
2. cgamex86.dll ：爲用戶端程式，負責處理用戶端邏輯、通訊介面與狀態顯示等。
3. uix86.dll：爲用戶端顯示介面，包含選單、大文字與滑鼠操控等。

- Half-Life 三維電腦遊戲引擎

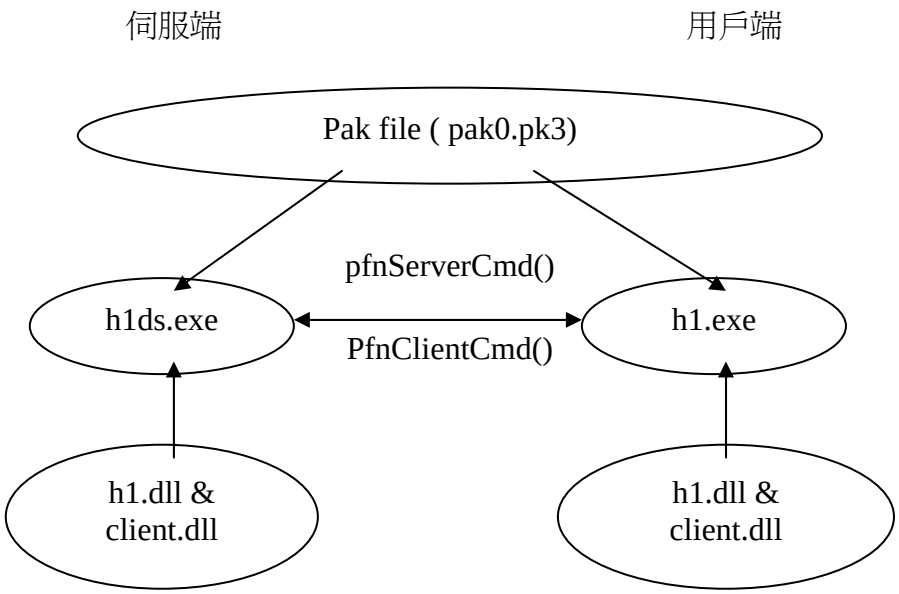


圖 6　Half Life 三維電腦遊戲引擎的軟體執行架構

1. h1s.exe ：伺服端的執行程式。
2. h1.exe ：用戶端的執行程式。
3. h1.dll ：爲三維電腦遊戲引擎的核心程式模組，負責所有實體的運作。
4. client.dll ：包含伺服端與用戶端的操作介面程式模組。.

- Fly3d 三維電腦遊戲引擎

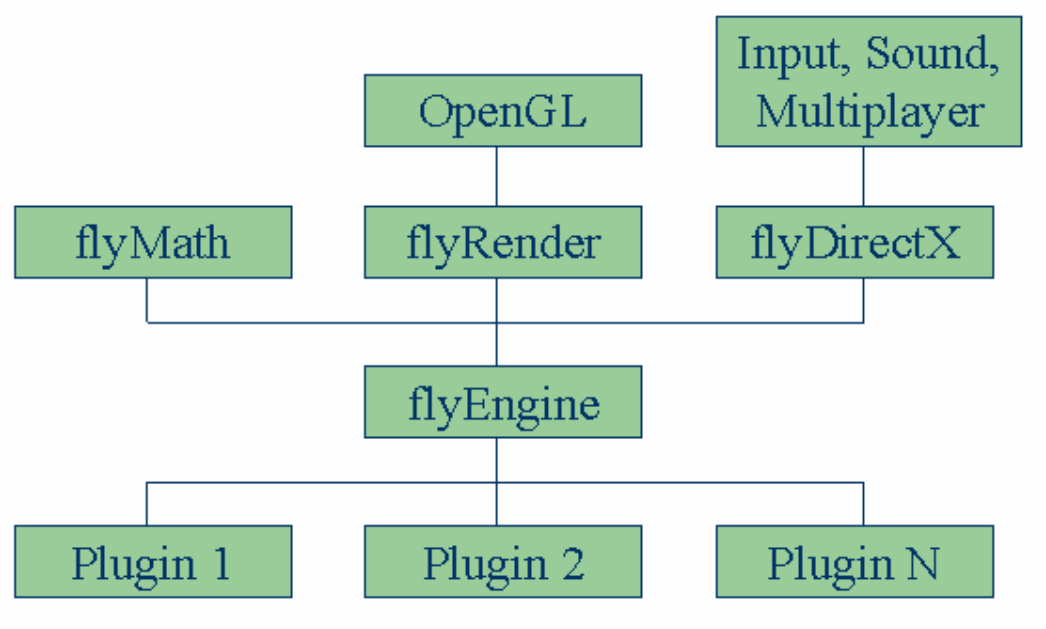


圖 7　Fly3d 三維電腦遊戲引擎的軟體架構

1. flyMath　：數學函數程式模組。
2. flyRender ：OpenGL 繪圖程式模組，包含我們撰寫的 NPR 程式模組。
3. flyDirectX：微軟的 DirectX 介面程式模組。
4. Plugin　　：系統提供與自行發展的物件處理程式模組。

1.4 章節概要

本論文架構共分七章，其內容概述如下：

- 第一章：介紹中國水墨人物畫之發展史、說明本論文的研究動機與目標、系統架構與三維電腦遊戲引擎之評估。

- 第二章：介紹NPR之相關研究。

- 第三章：繪圖演算法，包含水墨毛筆模組之模擬機制與NPR的繪圖處理步驟。

- 第四章：繪圖狀態機，包含動畫場景的繪圖狀態機與靜態場景的繪圖狀態機。

- 第五章：電腦象棋的介紹與人工智慧演算法。

- 第六章：顯示實作結果。

- 第七章：結論。

二、 相關研究

根據不同的繪圖風格，在NPR的研究上可分成幾個不同的方向，例如：以西畫爲主的水彩繪[5]、影線畫[8, 19]、鉛筆畫[30, 31]與及剛筆墨畫[27, 28, 29]等，另以中國水墨畫爲主的岩石水墨畫[33]、樹木水墨畫[34]及動物水墨畫[36] 等。而本論文研究目標爲將中國水墨風格應用的三維電腦遊戲引擎上，以呈現中國水墨風格的遊戲場景，與動物水墨畫[36]有些相似。

我們知道模型輪廓線與特徵線的萃取，在NPR的繪圖處理上是相當重要的。其萃取的方法約略分爲三類：影像空間演算法（Image Space）、物件空間演算法（Object Space）與混合演算法（Hybrid）[22]。物件空間演算法有下列幾種：根據定義來偵測的暴力法（Brute Force，爲我們使用的方法）、特殊資料結構的Edge Buffer [4]、機率法[19]、階層式高斯弧形圖[10]（Gauss Map Arc Hierarchy）、階層式標準圓錐體[16]（Normal Cone Hierarchy）、隱含式曲面[15]（Implicit Surfaces）與NURBS曲面[6]等。而影像空間演算法有下列幾種：單次法（One Pass）[7,8]、雙次法（Two Pass）[10,24]、環境圖法[10]（Environment Map）、模型參數法[19,26]（Model Augmentation）與不連續深度法[21]（Depth Discontinuity）等。最後是Northrup et al. [22]提出的混合演算法，首先他們使用影像空間演算法去找出可見的線段路徑，並將之連結成較長的線段，然而這些線段已具有物件空間的精確邊線，而且可避免產生鋸齒狀的輪廓線及與原始可見的輪廓線不一致

現象，其繪圖速度爲互動的等級。

另一個相當重要的演算法是可見範圍的選擇（Visibility Culling），亦就是去除隱蔽線演算法（Hidden Line Removal）。Appel's [3] 提出的除隱蔽線演算法是基於不可見數量的概念來處理（Quantitative Invisibility QI），他是去計數物件與視點位置之間正面（Front-Facing）的多邊形個數，作爲隱蔽線的判斷依據。Markosian et al. [19]修改前者提出新的演算法，藉由他的特殊遊走機制，不需要遊走測試所有的模型邊線，就可以偵測出可見線段及三維模型的可見曲面，最後再應用許多的線段風格於輪廓線與特徵線上。事實上最簡單也最快的是影像空間中的硬體 Z-Buffer 偵測方法，但由於影像點（Pixel）的精確度與部分遮蔽問題，使得 Z-Buffer 有些限制，其爲主要缺點。Kowalski et al. [20, 22]以唯一的顏色值，繪製輪廓線與三角面到後端的視框緩衝區（Frame Buffer）中，作爲 ID 參考影像（ID Reference Image），等全部繪製完後再從後端的視框緩衝區中，讀出 ID 參考影像到主記憶體中做可見性測試（Visibility Test）。Isenberg et al. [14] 使用上述相同的方法，不同的是他們的可見性測試，是使用 Z-Buffer 來代替 ID 參考影像。本論文中我們使用 ID 參考影像來處理遮蔽線問題，因爲 ID 參考影像可以識別出可見的輪廓線與可見的三角面，如此便能偵測出模型的特性部分。

另外有關繪畫風格的模擬，Curtis et al. [5] 提出了以媒體的物理特性爲基礎的二維影像處理方式來模擬水彩畫，而他們的研究以二維影像處理爲主，而沒有將三維模型的資訊應用於三維的場景繪製上。另一個重要的NPR的藝術風格是鋼筆墨畫[27, 28, 29]，而大部份擁有鋼筆墨畫風格的研究，主要還是依靠著三維模型的參數性質爲研究重點。另一項與鋼筆墨畫研究相關的是鉛筆畫研究[30,31]，Praun et al. [23]將鉛筆畫的風格應用到三維的摸型上，他的研究主要貢獻是：色調圖，當材質的顏色色調改變時，可以避

免不連續的現象，此項色調圖（Tonal-Maps）研究的靈感是來自Klein et al. [17]的虛擬NPR環境之研究（Non-Photorealistic Virtual Environments）。

筆刷的模擬研究方面，有 Strasmann [32]提出的：以一維陣列當作繪畫的筆刷，而陣列的元素存放著筆毛個別的墨量值、Horace and Helena [12]提出的：藉由三維立體圓錐的物理性質與動態的移動方式，模擬中國書法風格的三維毛筆模組與 Weng et al. [35]提出的：定義以一個平面圓形，作爲筆刷與畫布的接觸區域，與均勻分配於該接觸區域內的筆毛數量，模擬中國水墨風格的二維毛筆模組。然而 Weng 的二維毛筆模組模擬的結果看起來較自然且平滑，因此我們參考 Weng 的二維毛筆模組，應用於我們的系統上，其實作容易且視覺效果好。

三、 繪圖演算法

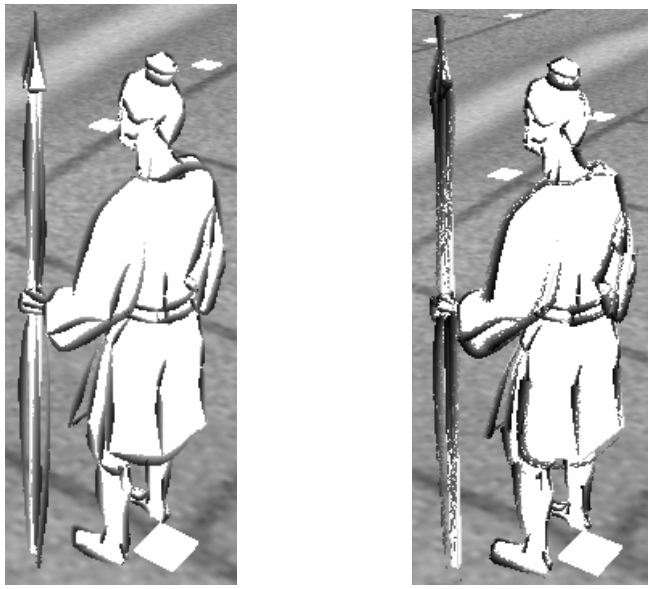
本章說明我們的繪圖演算法，以模擬出中國人物白描的水墨效果。下面我們會介紹水墨畫模組中的毛筆模組與移動控制機制、水墨筆觸模擬與濃淡效果、水墨特性與毛筆壓力機制。在三維模型資訊萃取處理步驟上，我們介紹如何找出輪廓線與特徵線、如何去除隱蔽線、將較小可見線段連結成較長的線段與產生控制點。最後介紹我們模擬的各種白描技法。

3.1 水墨畫模組

3.1.1 毛筆模組與移動控制機制

本節將介紹以 Weng 爲主的二維毛筆模組， Weng 的毛筆模組是二維圓形的架構來模擬毛筆與畫布接觸區域，並著重在中鋒的筆觸模擬上，然而我們可以藉由將二維圓形改變成二維橢圓的架構，去模擬毛筆與畫布接觸區域的墨水量多寡，以達到模擬側鋒的效果，圖 8 顯示毛筆中鋒與側鋒的模擬範例。我們以同心橢圓的分布方式來表示毛筆與畫布的接觸區域，如圖 9 所示，以 O 爲中心點的同心橢圓來表示，並以最外圈（$\boldsymbol{F_I}$）作爲基準，內圈的橢圓則以比例方式分佈，我們定義 Y 軸的方向爲主軸（Q），作爲軸長比率的基準，而 X 軸的方向爲副軸（P）。其個別的橢圓方程式與軸長定義如 3-1 式，

參數式如 3-2 式。



(a) (b)

圖 8 毛筆中鋒與側鋒的模擬範例(a)中鋒，(b)側鋒

$$F_i \Rightarrow \frac{x^2}{p_i^2} + \frac{y^2}{q_i^2} = 1$$

$$p_i = P \times \frac{i}{I}$$

$$q_i = Q \times \frac{i}{I} \qquad (3-1)$$

F_i 的橢圓參數式如下所示：

$$x_i = p_i \cos\theta$$

$$y_i = q_i \sin\theta \qquad \theta \in [0, 2\pi] \qquad (3-2)$$

其中：

p_i , q_i , P , Q 分別爲 F_i 與 F_I 的主軸與副軸，

I 爲橢圓的個數，

i 爲內部所屬的橢圓編號。

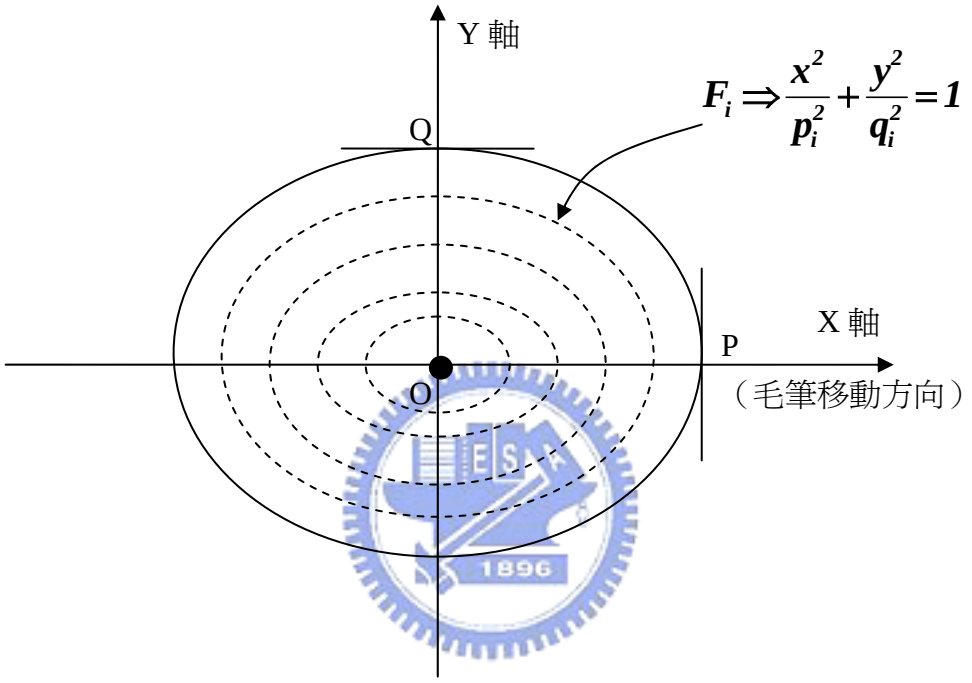


圖 9 毛筆與畫布接觸區域之定義

我們定義筆毛與畫布接觸的線段，稱作區域 R_j，橢圓以等角度方式切割成數個小區域，如圖 10 所示，因此線段與 X 軸的夾角可由 3-3 式求得。

$$\theta_j = 2\pi \times \frac{j}{J} \qquad (3-3)$$

其中

$0 \le \theta_j < 360$ 爲 X 軸與 $\overline{OS_{i,j}}$ 的夾角。

J 爲分割的區域的總數 ，

j 爲區域的編號。

對於F_i 與 R_j的交點$S_{i,j}$位置，我們可以極座標方式表示如3-4式：

$$S_{i,j} = S\left(d_{i,j}, \theta_{i,j}\right) \qquad (3-4)$$

其中

i 爲橢圓的編號，

j 爲分割區域的編號，

$d_{i,j}$ 爲 $S_{i,j}$ 到原點 O的距離 (畢氏定理)。

$$d_{i,j} = \sqrt{\left(P \times \frac{i}{I} \times \cos\theta_j\right)^2 + \left(Q \times \frac{i}{I} \times \sin\theta_j\right)^2} \qquad (3-5)$$

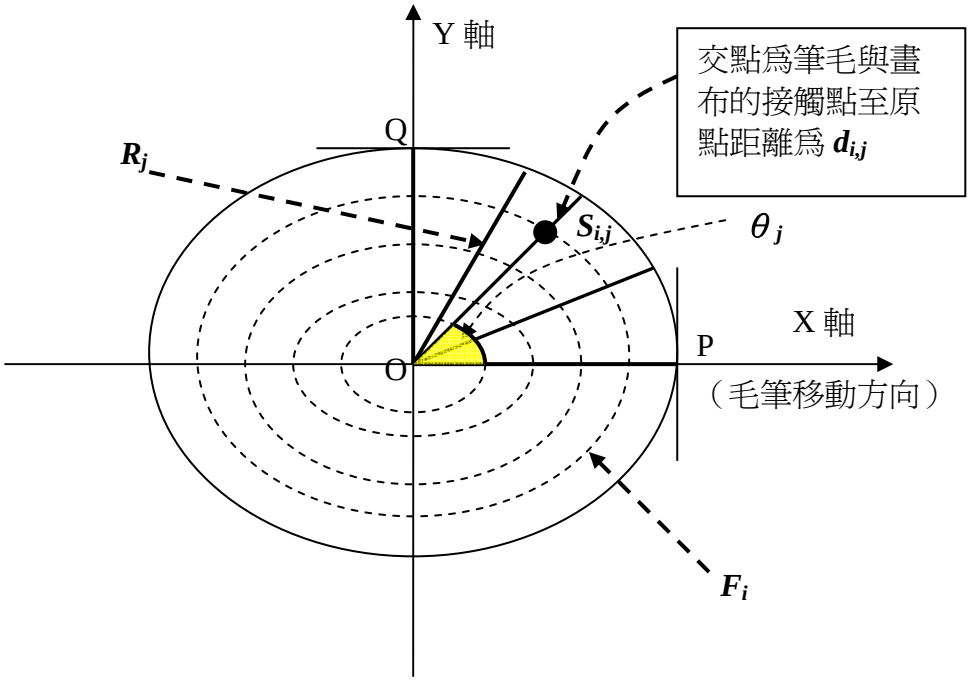


圖 10 以等角切割成數個區域

毛筆模組中的***P*** 、***Q*** 參數可以控制整個毛筆與畫布接觸的形狀與面積，在我們的系統中副軸***P*** 是以主副軸的比例***R*** 來代替（***P*** = ***R*** x ***Q***），我們可以改變***Q*** 的大小決定毛筆與畫布接觸的範圍，及改變***R*** 的大小來決定毛筆與畫布接觸面的外型。筆毛的分布情形，則由***I*** 橢圓的個數與***J*** 分割的區域的總數來控制。由此我們藉由改變***Q***、***R***、***I*** 與***J*** 四個參數以模擬出中鋒與側鋒的不同筆觸。

我們根據控制點以***Cardinal Spline*** 建立毛筆移動軌跡，移動軌跡曲線的控制點則以內插方式計算獲得，其移動過程包含毛筆位置、毛筆壓力及毛筆軸向，說明如下。

1. 毛筆位置：

假設毛筆中心點 ***O***在畫布 ***(x , y)*** 位置上 **,** 則毛筆與畫布的接觸點 $S_{i,j}$

定義如下：

$$S_{i,j,x} = S'_{i,j,x} + x = (P \times \frac{i}{I} \times \cos\theta_j) + x$$
$$S_{i,j,y} = S'_{i,j,y} + x = (Q \times \frac{i}{I} \times \sin\theta_j) + y \qquad (3 - 6)$$

其中

$S'_{i,j}$ 爲 $S_{i,j}$ 在毛筆模組的區域座標 位置，

$S_{i,j,x}$, $S_{i,j,y}$, $S'_{i,j,x}$, $S'_{i,j,y}$ 分別爲 $S_{i,j}$ 與 $S'_{i,j}$ 的 ***x , y*** 座標位置。

2. 毛筆壓力：

假設有***n+1***個曲線控制點 c_1 , c_2 , c_3 , …… , c_n , c_{n+1} , 將曲線

分成***n***個線段 s_1 , s_2 , s_3 , …… , s_n 。

則在某一線段s_i中的內插點d，可以下式求得該點壓力值：

$$\frac{p_d - p_{c_i}}{p_{c_{i+1}} - p_{c_i}} = \frac{u_d - u_{c_i}}{u_{c_{i+1}} - u_{c_i}} \qquad (3 - 7)$$

其中

$0 \le p_d , p_{c_i} , p_{c_{i+1}} \le 1$ 分別爲 d , c_i , c_{i+1} 各點的壓力值，

$0 \le u_d , u_{c_i} , u_{c_{i+1}} \le 1$ 分別爲d , c_i , c_{i+1} 各點的***Cardinal Spline*** 參數。

3. 毛筆軸向：

我們參考Weng[35]利用毛筆軸向對齊曲線的切線方法，以獲得較自然的筆觸。如圖11所示，毛筆沿著軌跡移動，在第***k***個軌跡點d_k，利用d_k點上的斜率可以計算出切線向量T_k與X軸的夹角如***3-8***式所示：

$$T(x_k, y_k) = \frac{y_k - y_{k-1}}{x_k - x_{k-1}}$$

$$\alpha_k = Tan^{-1}(T(x_k, y_k)) \qquad (3-8)$$

其中

$T(x_k, y_k)$ 爲切線向量 T_k 的斜率，

α_k 爲T_k與X軸的夹角。

將第k個軌跡點的 α_k值減去前一個軌跡點的 α_{k-1}，可以得到 d_{k-1}到d_k的毛筆旋轉角度 α，毛筆旋轉後的新座標爲 ：(三角和差化積公式)

$$S_{i,j,x} = (P \times \frac{i}{I} \times \cos\theta_j \times \cos\alpha - Q \times \frac{i}{I} \times \sin\theta_j \times \sin\alpha) + x$$

$$S_{i,j,y} = (P \times \frac{i}{I} \times \cos\theta_j \times \sin\alpha + Q \times \frac{i}{I} \times \sin\theta_j \times \cos\alpha) + y \qquad (3-9)$$

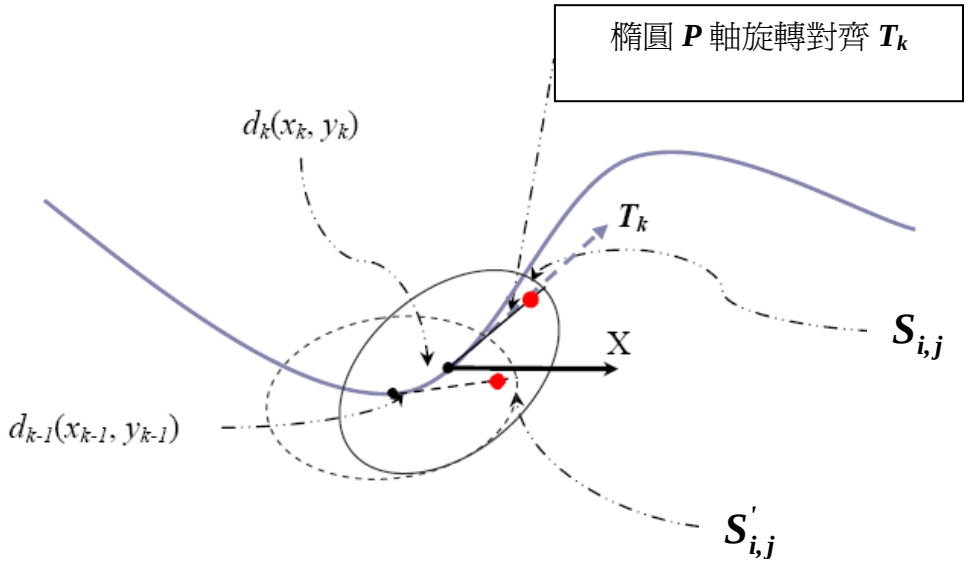


圖 11 毛筆的移動軌跡與旋轉

3.1.2 水墨筆觸模擬機制

3.1.2.1 墨色變化

在Weng[35]的毛筆模組中，是將墨水量與墨色視爲相同，使得無法模擬在墨色濃水量少的情形，因此我們將墨色值與墨水量分開。我們將Weng毛筆模組中的Decreasing 參數模擬墨水量減少產生的墨色變化，更改成以水與墨的濃度不同產生的墨色變化，如圖12所示。爲了模擬毛筆所畫的墨色深淺濃淡變化，我們加入了墨色控制參數，將筆觸的起始墨色與終止墨色作一限制，以線性內插的方式模擬筆觸的墨色變化情形。

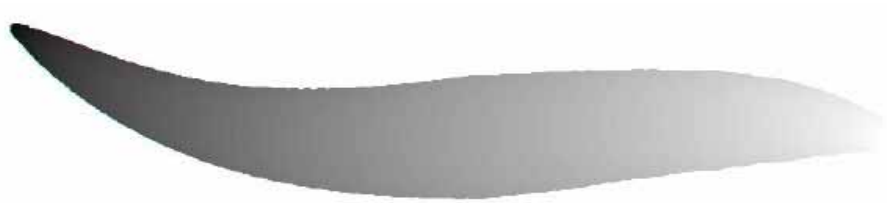


圖 12 墨色變化效果

我們參考了Weng毛筆模組中的Concentration、Difference及Discontinuity三個參數，利用前兩個參數來控制墨色及墨水量的分布，第三個參數則用於模擬墨水量不足時產生斷水效果，說明如下：

1. 水墨集中效果（Concentration）

Weng在此效果上提供了四種墨色分布，他的墨色分布效果是在毛筆與畫布的接觸面Y方向上做出線性的灰階變化，如圖13所示，此效果可與其他參數配合產生不同的筆觸風格。

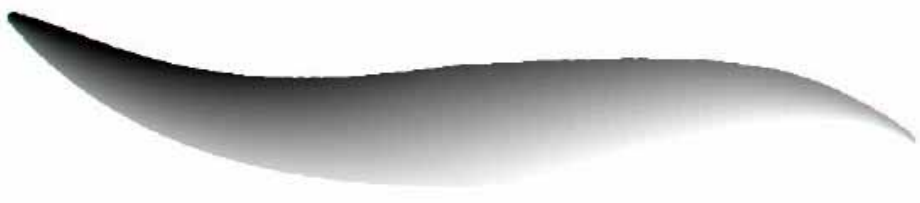


圖 13 墨色在Y方向作線性灰階變化產生的效果

2. 筆毛水墨相異值（Difference）

在這個效果上Weng以一個亂數陣列（difference[]），存放著由從正負相同區間內取出的亂數值，再以筆毛的y值對亂數陣列索引，以取得不同的亂數值，如此根據不同的筆毛y值就可以模擬出不同的墨色擾亂程度，其效果如圖14所示。

圖 14 不同的墨色擾亂程度產生的效果

3. 筆毛斷水效果（Discontinuity）

在繪圖的時候，毛筆所沾上的墨水量會隨著筆毛與畫布的接觸移動逐漸減少，爲模擬該項效果，定義了不連續陣列（discontinuity[]），與亂數陣列（difference[]）作法相同，其索引出來的數值就作爲筆毛發生斷水的長度上限（Max_Gap）。當筆毛的墨水量不足時，亂數產生小於等於Max_Gap的斷水長度gap-size，讓筆毛於該區段距離內不留下墨跡，其效果如圖15所示。

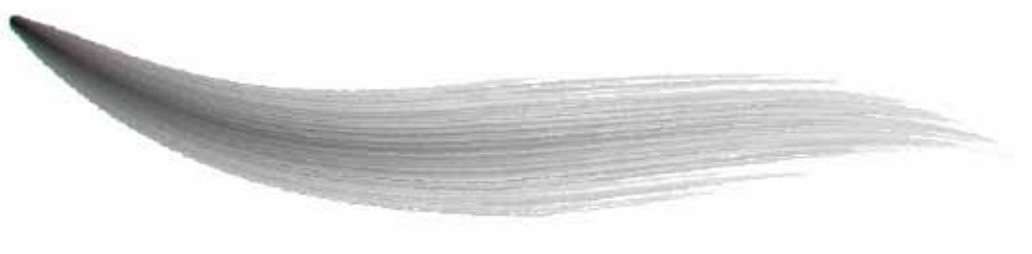


圖 15 筆毛斷水效果

3.1.2.2 中鋒與側鋒模擬

中鋒移動方式爲直筆而下，以影響毛筆與畫布接觸面的大小及墨水滲出量，然而接觸面的大小及墨水滲出量與毛筆壓力有關，如圖16所示，而墨色的深淺代表著每根筆毛滲出墨水的加權數W_p高低，以3-10定義如下：

$$W_p = \begin{cases} no\ touch & , \dfrac{d_{i,j}}{d_{I,j}} > p \\ \sqrt{((1 - \dfrac{d_{i,j}}{d_{I,j}}) \times p + p)/2} & , \dfrac{d_{i,j}}{d_{I,j}} \le p \end{cases} \qquad (3 - 10)$$

其中

$d_{i,j}$ 爲$S_{i,j}$ 到橢圓中心點 O 的距離，
$d_{I,j}$ 爲$S_{I,j}$ 到橢圓中心點 O 的距離，
$0 \le p \le 1$ 爲毛筆的壓力值。

由*3 - 5*式可以得出：

$$d_{i,j} = d_{I,j} \times \frac{i}{I} \qquad (3 - 11)$$

因此 *3 - 10*可簡化爲：

$$W_p = \begin{cases} no\ touch & , \dfrac{i}{I} > p \\ \sqrt{((1 - \dfrac{i}{I}) \times p + p)/2} & , \dfrac{i}{I} \le p \end{cases} \qquad (3 - 12)$$

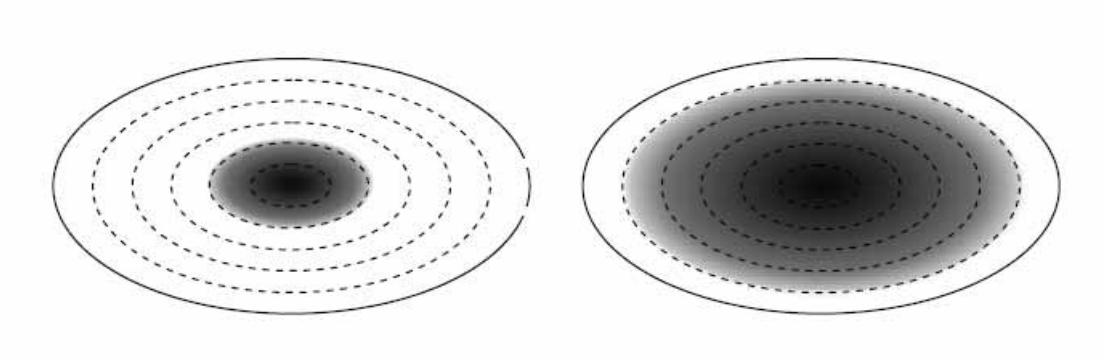


圖 16　毛筆壓力與接觸面積成正比

側鋒移動方式爲斜筆而下，提筆時毛筆與畫布接觸面便由單邊開始逐漸往筆尖縮小，如圖 17 所示，其筆毛滲出墨水的加權數 W_p 高低，以 3-13 定義如下：

$$W_p = \begin{cases} no\ touch & , \dfrac{S_{i,j,y} + Q}{2Q} > p \\ p & , \dfrac{S_{i,j,y} + Q}{2Q} \le p \end{cases} \qquad (3\text{-}13)$$

其中

$S_{i,j,y}$ 爲 $S_{i,j}$ 的 Y 座標值。

移動方向

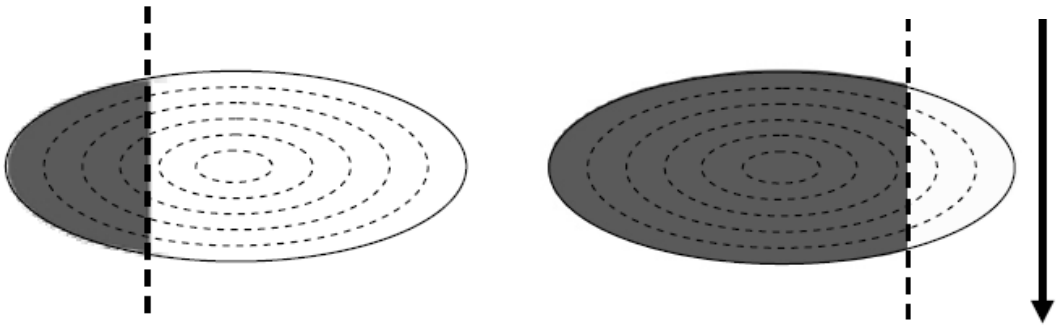


圖 17 側鋒接觸面由單邊提筆

3.2 三維模型資訊萃取

本章節將說明我們應用於系統中，與三維模型資訊萃取相關的方法與步驟，我們的方法是參考 Northrup et al. [22]提出的混合演算法與 Kowalski et al. [20]提出的 ID 參考影像（ID Reference Image），綜合如下所示：

1 三維空間處理步驟

1.1 找出模型的輪廓線與特徵線（稱之爲線段 Segment），及儲存線段投影至二維平面的相關資訊。

1.2 開啓 Z-buffer 功能處理物體遮蔽問題，繪製 ID 參考影像（ID Reference Image）。

2 二維空間處理步驟

2.1 讀取視框緩衝區（Frame Buffer），根據 ID 參考影像從新檢測，設定線段的可見（Visibility）與否，此設定將去除被其他物體遮蔽的線段。

2.2 處理可見線段的重疊與可能產生的小鋸齒現象。

2.3 將處理後的可見線段連結成較長的線段，並以 ***Cardinal Spline*** 產生較平滑的曲線，作爲毛筆移動的軸線。

2.4 沿著曲線模擬毛筆水墨筆觸。

3.2.1 找出輪廓線與特徵線

有關 NPR 的研究，找出模型的輪廓線與特徵線是很重要的，如何找的快又找得正確，是件不容易的事，雖然有許許多多的方法用於解決此項問題，如前面所述有影像空間演算法（Image Space）、物件空間演算法（Object Space）與混合演算法（Hybrid）等[22]，其又個別有不同的方法，但仍離不開基本定義，如圖 18 所示說明如下：

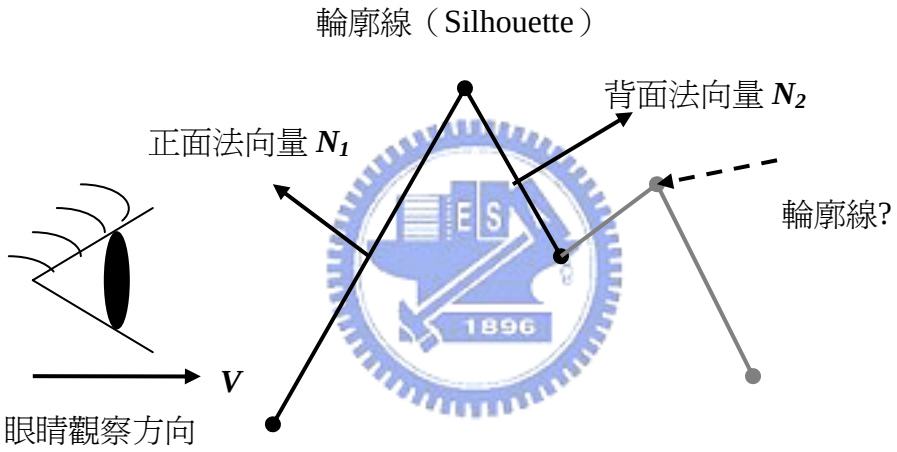


圖 18 輪廓線之定義

假設 $\boldsymbol{V}$ 爲眼睛的觀察方向，$\boldsymbol{N}$ 爲多邊形面的法向量（方向往外），則正面（Front-Face）與背面（Back-Face）之判斷如 3-15 式所示：

1. 如果 $V \bullet N < 0 \Rightarrow$ 爲正面，
2. 如果 $V \bullet N > 0 \Rightarrow$ 爲背面，
3. 如果 $V \bullet N = 0 \Rightarrow$ 垂直觀察方向。 **(3 - 15)**

上式中 $\bullet$ 表內積運算。

我們知道所謂的輪廓線是一個邊（Edge）同時爲鄰近的兩個多邊形面所共用，由 3-16 定義可偵測出輪廓線，我們將垂直觀察方向也包含近來，因爲在視覺上有可能是一個不可忽視的邊。

如果 $(V \bullet N_1) \times (V \bullet N_2) \leq 0 \Rightarrow$ 爲輪廓線 **(3 - 16)**

其中

V 爲眼睛觀察方向，

N_1, N_2 分別爲多邊行正面與背面的法向量，

上式我們把垂直的部份也包含進來。

接下來我們說明如何偵測特徵線，在特徵線的定義中可能包含了邊緣線（Boundary Edge）、摺痕（Crease Edge）與兩個不同材質所產生的材質邊線（Material Edge），我們將對前二種提出討論，第三種屬材質屬性問題不做討論，對於邊緣線我們在計算輪廓線時，可事先加以判斷即可，而摺痕則是由鄰近的兩個多邊形面產生凹陷的山谷（Valley）或凸起的山脊（Ridge）所構成，因此我們藉由計算其兩平面的夾角小於某個角度來獲得，通常是 60 度角，如 3-17 式所示，另外我們爲加快計算速度，實作上我們將照相機的座標位置反轉至模型的座標系統上，以減少座標系統的轉換次數。

如果 $\left|N_1 \bullet N_2\right| > 0.5 \Rightarrow$ 爲摺痕 ***(3 - 17)***

因***C***語言的***cos***函數傳回的是最小夾角的值故不需另作判斷。

3.2.2 去除隱蔽線

由圖 18 我們發現位於最右邊的輪廓線被其他物體遮擋，對觀察者而言是看不到的，然而就基本定義的計算上是可見的，因此本節將說明如何處理該項問題。首先我們依基本定義計算偵測出輪廓線與特徵線，並指定唯一的顏色值由 OpenGL 直接繪製成 ID 參考影像[20]（ID Reference Image）如圖 19 所示，並開啓 Z-Buffer 來解決物體遮擋問題，另外我們將偵測出輪廓線與特徵線記錄起來，紀錄的內容包括：兩端點座標位置、兩端點的二維平面投影值、輪廓線編號（唯一大於 128 的顏色值）及可能相鄰的輪廓線與特徵線等資訊，我們稱之爲線段（Segment）。

當所有的模型繪製完後，由後端的視框緩衝區中將 ID 參考影像讀至主記憶體，根據所有先前紀錄的可見線段兩端點的二維平面投影值，逐一掃描 ID 參考影像作線段可見確認，爲避免電腦數值表示與運算產生的誤差，我們以 3x3 像素（Pixel）爲掃描 ID 參考影像的範圍，對於可見線段的重疊與產生的小鋸齒現象，我們藉由實作毛筆筆觸時的加以覆蓋來消除，然而重疊的筆觸部份更可表現出墨水較濃的效果。

圖 19 被選擇的棋子與場景的 ID 參考影像（爲加快執行速度）

3.2.3 線段連結與產生控制點

作完上面的可見線段確認後，接著我們根據每一個可見線段的兩端點的二維平面投影值，來搜尋 ID 參考影像，並以可見線段之間最大 45 度的夾角與最大 5x5 的像素，作爲 ID 參考影像的搜尋範圍，將找出的可能相鄰可見線段加以紀錄，最後我們再將每一個可見線段與紀錄的相鄰可見線段連結成較長的線段。演算法如圖 20 所示：

Link Segment(S_1 , S_2)

θ_{max}: 允許的最大連結角度
G_{max}: 允許的最大連結端點間隙

for 所有的線段 S_1
for 所有的線段 S_2
{
θ ： 線段 S_1 與 S_2 的夾角
G ： 線段 S_1 與 S_2 之間的端點間隙
if (S_1 已連結到 S_2 時) ***or*** ($\theta \geq \theta_{max}$)
排除 S_2
if (線段 S_1 與 S_2 之端點不重疊)
and ($G \leq G_{max}$)
and ($\theta \leq S_1$的目前任意相鄰線的夾角)
and ($\theta \leq S_2$的目前任意相鄰線的夾角)
將 S_1 與 S_2 連結
}

圖 20 線段連結演算法

連結成較長的線段後，接下來是產生控制點，我們應用的是 ***Cardinal Spline*** 如 3-18 式所示，***Cardinal Spline*** 的優點是：以內插的方法產生曲線，並且會經過我們所指定的控制點，其精確度高且較具彈性。我們以可見線段的端點作爲 **Cardinal Spline** 的控制點，並藉由調整 ***Cardinal Spline*** 的張量（Tensor）參數 ***t*** 以改變曲線的柔軟效果如圖 21 所示，最後我們再沿著曲線模擬毛筆水墨筆觸，以呈現水墨風格。

假設曲線 $p(u)$ 位於控制點 p_k 與 p_{k+1} 之間，則另需前後兩個 控制點 p_{k-1} 與 p_{k+2} ，其表示式如下：

$$p(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot M_c \cdot \begin{bmatrix} p_{k-1} \\ p_k \\ p_{k+1} \\ p_{k+2} \end{bmatrix} \qquad (3-18)$$

其中 ***cardinal*** 矩陣爲：

$$M_c = \begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -s \\ -s & 0 & s & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad (3-19)$$

上式 $s = (1-t)/2$。

$$\Rightarrow p(u) = p_{k-1}(-su^3 + 2su^2 - su) + p_k\left[(2-s)u^3 + (s-3)u^2 + 1\right] + p_{k+1}\left[(s-2)u^3 + (3-2s)u^2 + su\right] + p_{k+2}(su^3 - su^1) \qquad (3-20)$$

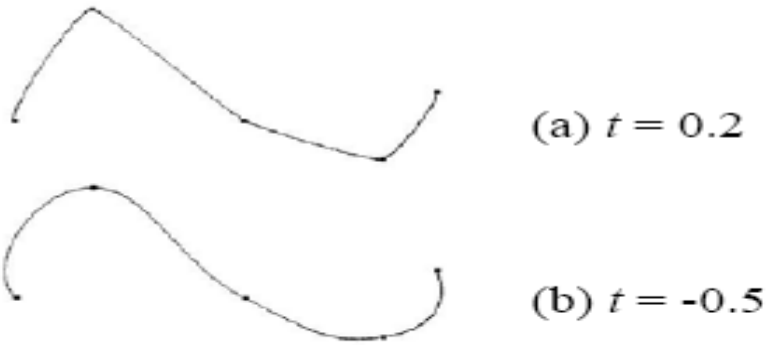


圖 21 不同參數 t 的 ***Cardinal Spline***

3.3 白描技法模擬

我們於第一章介紹過，中國水墨人物白描技法有十八種之多，依線條之變化可分類爲無粗細變化（鐵絲描類）、有粗細變化（蘭葉描類）及快速減化筆線（減筆描類）等三種，本節將說明我們模擬的無粗細變化的琴弦描、有粗細變化的柳葉描與棗核描及快速減化筆線的減筆描與竹葉描，最後是前述的綜合技法模擬。

在白描技法的模擬上，主要是以線條的粗細變化來寫意的，因此我們以改變毛筆模組中的壓力參數 ***p*** ，及配合連結的線段長度與毛筆模組的控制機制，模擬出中國水墨人物白描技法的效果，說明如下：

1. 無粗細變化的琴弦描：此白描技法下垂時有如琴弦一般長而平行的線好幾條向同一方向伸去，下筆時力量均勻而細勁，沒有明顯起伏波折。我們根據連結的線段長度，設定不同的橢圓主軸與主副軸比率大小，及以固定的壓力值來模擬。

2. 有粗細變化的柳葉描與棗核描：柳葉描筆意比較柔軟，重撇後輕輕轉折如柳葉，我們根據連結的線段長度，設定不同的橢圓主軸與主副軸比率大小，並以壓力值由 0→ 1→ 0 的變化來模擬，其中 0→1 與 1→0 的線段長度以亂數決定，使產生的效果不致太過規律。棗核描用尖筆藏峰，頓挫如棗核，我們只對長度在 20→ 40 之間的連結線段作棗核描模擬，依據長度不同設定其橢圓的主軸與主副軸比率大小，及將該線段分成若干等分，每等分以壓力值由 0→ 1→ 0 的急速變化模擬。]

3. 快速減化筆線的減筆描與竹葉描：兩著相似，比柳葉描更爲粗短，像大筆撇竹葉一般，線條較硬下筆後急速提筆，我們根據連結的線段長度，設定不同的橢圓主軸與主副軸比率大小，並以壓力值由 1→ 0 的急速變化模擬。我們以中鋒模擬減筆描，而竹葉描則以側鋒模擬之。

4. 前述的綜合技法模擬：我們依連結的線段長度，分別依前述的柳葉描、棗核描與減筆描模擬之。

四、 繪圖狀態機

我們知道在 NPR 的繪圖研究上，繪圖的速度是個重要關鍵因素，主要是因爲輪廓線會隨著視點改變而跟著需要重新計算，因此在傳統的繪圖機制上每重畫一次視框就得重新計算一次，導致 CPU 的負載加重使得繪圖的速度變慢，通常維持在互動的速率上（Interactive Rate），對於 NPR 的繪圖速度提昇上我們提出繪圖狀態機[25]的機制，企圖用於提昇三維電腦遊戲引擎對輸出入裝置之反應速度，使畫面顯示與遊戲之操作得以更加順暢。下面我們分別說明動畫繪圖狀態機、靜態繪圖狀態機與兩著的效果分析。

4.1 動畫繪圖狀態機

所謂的動畫繪圖狀態機，是針對動畫的場景做分類的，通常在動畫的場景中每次會變動的物體爲場景中較少部分，因此我們根據此項特性，將傳統的繪圖機制拆成四個狀態分別是：全部繪製（並儲存影像）、先繪靜態部份（並儲存影像與深度值）再繪動畫物件、回存靜態部份深度值再繪動畫物件與將儲存影像直接輸出至螢幕等四個不同繪圖狀態，狀態的改變會依據系統目前所處的狀態及其變化情形，選擇下一個繪圖狀態來繪圖，下面我們說明狀態圖、真值表與導出的邏輯式子，如圖 22、表 1 至表 4 及 4-1 式所示：

假設繪圖機制分成四個狀態分別如下：

A **：** 全部繪製**(**並儲存影像**)**，
B **：** 先繪製靜態部份**(**並儲存影像與深度值**)**，再繪製動畫物件，
C **：** 回存靜態部份深度值，再繪製動畫物件，
D **：** 將儲存影像直接輸出至螢幕。

另外影響狀態變化的變數如下：

x_0 **：** 棋子被選取與否 **(** $0 \Rightarrow$ 未被選取，$1 \Rightarrow$ 被選取**)**，
x_1 **：** 視點改變與否 **(** $0 \Rightarrow$ 未改變，$1 \Rightarrow$ 已改變**)**，
x_2 **：** 到達目的位置與否 **(** $0 \Rightarrow$ 未到達，$1 \Rightarrow$ 已到達**)**，
Φ **：** 不會發生的狀態 **(*don' t case*)**

其中的條件限制如下：

1. 當棋子移動時，需到達目的地後才能改變狀態。
2. 當視點改變時**(**包含取消選取**)**無論處於何種狀態，都需回到***A***狀態。

狀態轉換說明：

1. 狀態***A*** $\Rightarrow$ 當視點改變時$x_1 = 1 \rightarrow$ 狀態***A***；
當視點不變$x_1 = 0$ 且棋子未被選取$x_0 = 0 \rightarrow$ 狀態***D***；
當視點不變$x_1 = 0$ 且棋子被選取$x_0 = 1 \rightarrow$ 狀態***B***。
2.狀態***B*** $\Rightarrow$ 當視點改變時$x_1 = 1 \rightarrow$ 狀態***A***；
當視點不變$x_1 = 0 \rightarrow$ 狀態***C***。
3.狀態***C*** $\Rightarrow$ 當視點改變時$x_1 = 1$ 或 到達目的位置$x_2 = 1 \rightarrow$ 狀態***A***；
當視點不變$x_1 = 0 \rightarrow$ 狀態***C***。
4.狀態***D*** $\Rightarrow$ 當視點改變時$x_1 = 1 \rightarrow$ 狀態***A***；
當視點不變$x_1 = 0$ 及棋子未被選取$x_0 = 0 \rightarrow$ 狀態***D***；
當視點不變$x_1 = 0$ 且棋子被選取$x_0 = 1 \rightarrow$ 狀態***B***。

1. 狀態圖

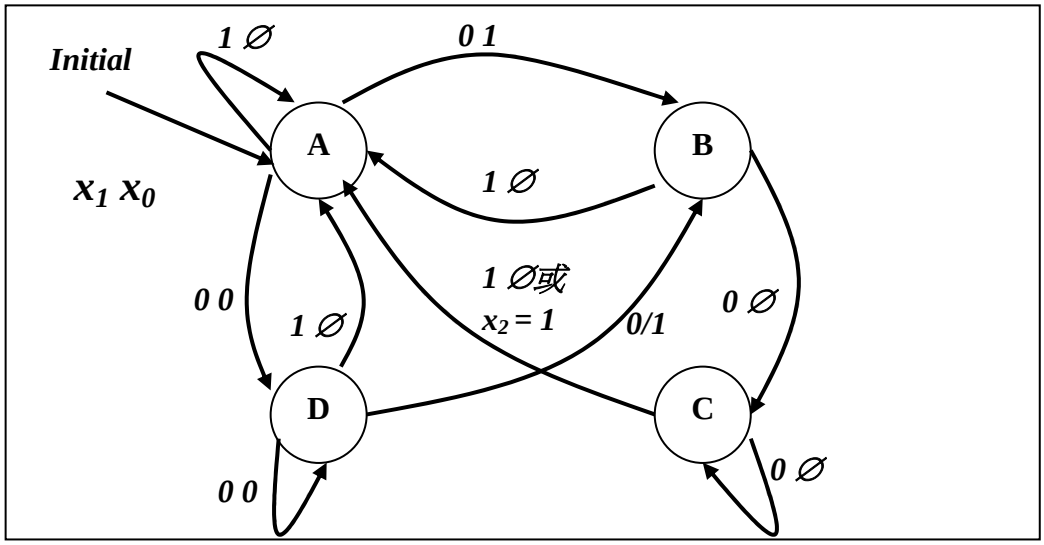


圖 22 動畫繪圖狀態圖

2. 真值表

其中 Q_n 表示目前狀態，Q_{n+1} 表示下個狀態。

表 1 動態狀態 A 的真值表

Q_n	x_1	x_0	Q_{n+1}
A	0	0	D
A	0	1	B
A	1	0	A
A	1	1	A

表 2 動態狀態 B 的真值表

Q_n	x_1	x_0	Q_{n+1}
B	0	0	C
B	0	1	C
B	1	0	A
B	1	1	A

表 3 動態狀態 C 的真值表

Q_n	x_1	x_0	Q_{n+1}
C	0	0	C
C	0	1	C
C	1	0	A
C	1	1	A
$x_2 = 1 \rightarrow A$			

表 4 動態狀態 D 的真值表

Q_n	x_1	x_0	Q_{n+1}
D	0	0	D
D	0	1	B
D	1	0	A
D	1	1	A

3. 邏輯式子

$$A_{n+1} = x_1 + x_2$$
$$B_{n+1} = A\overline{x_1}x_0 + D\overline{x_1}x_0 = (A+D)\overline{x_1}x_0$$
$$C_{n+1} = B\overline{x_1} + C\overline{x_1} = (B+C)\overline{x_1}$$
$$D_{n+1} = A\overline{x_1}\,\overline{x_0} + D\overline{x_1}\,\overline{x_0} = (A+D)\overline{x_1}\,\overline{x_0} \qquad (4-1)$$

4.2 靜態繪圖狀態機

假如場景中無動畫的物件時，我們可以簡化成全繪與不繪兩個狀態，但爲配合上述的動畫繪圖狀態機，我們取動畫繪圖狀態機四個狀態中的，全部繪製（並儲存影像）與將儲存影像直接輸出至螢幕二個繪圖狀態，作爲靜態繪圖狀態機的狀態，狀態定義如 4-1 節所示，狀態圖、真值表與導出的邏輯式子說明如圖 23、表 5、表 6 及 4-2 式所示：

狀態轉換說明：

1. 狀態$A \Rightarrow$ 當視點改變時$x_1 = 1 \rightarrow$狀態A；

當視點不變$x_1 = 0 \rightarrow$狀態D。

2. 狀態$D \Rightarrow$當視點改變時$x_1 = 1 \rightarrow$狀態A；

當視點不變$x_1 = 0 \rightarrow$狀態D。

1．狀態圖

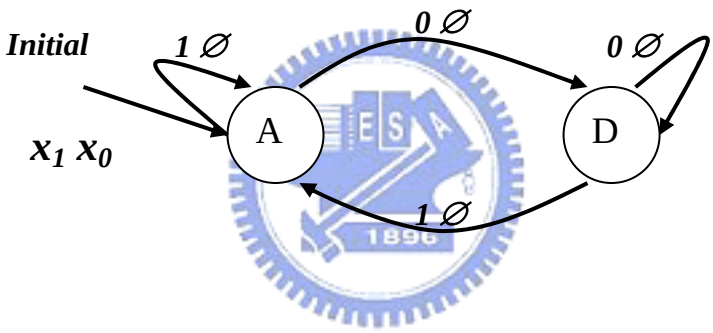


圖 23 靜態繪圖狀態圖

2．真值表

表 5 靜態狀態 A 的真值表

Q_n	x_1	x_0	Q_{n+1}
A	0	0	D
A	0	1	D
A	1	0	A
A	1	1	A

表 6 靜態狀態 D 的真值表

Q_n	x_1	x_0	Q_{n+1}
D	***0***	***0***	***D***
D	***0***	***1***	***D***
D	***1***	***0***	***A***
D	***1***	***1***	***A***

3．邏輯式子

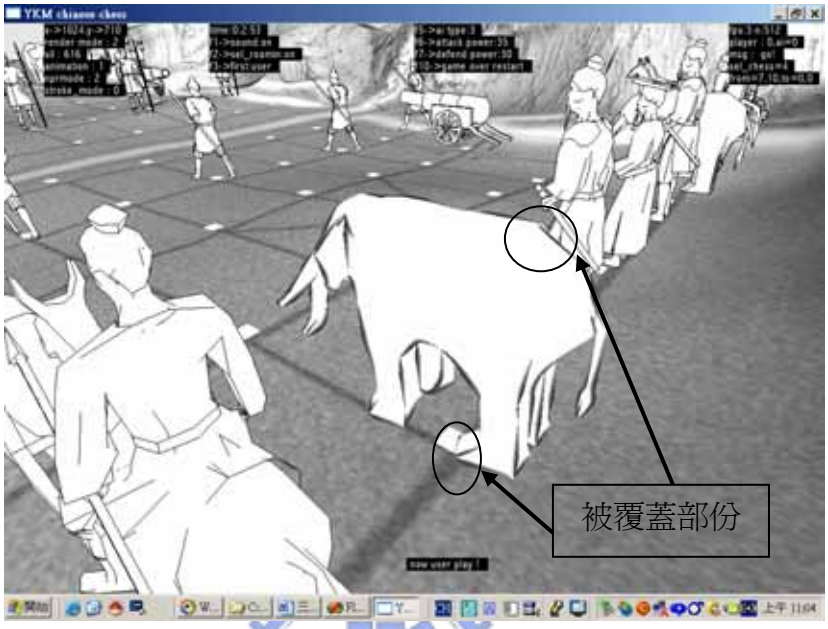
$$A_{n+1} = x_1$$
$$D_{n+1} = \overline{x_1} \qquad (4 - 2)$$

4.3 效果分析

我們提出的動畫繪圖狀態機是根據場景中變動的部分通常較少，此項特性來分出四個狀態，然而這四狀態中的回存靜態部份深度值再繪動畫物件狀態，有一個問題需要克服，就是除了回存靜態部份的深度值（Depth value）於 Z-Buffer，仍需於繪製完動畫物件後，再掃描 Z-Buffer 中的值是否被改變，若被改變則表示該點的靜態影像需爲該點的動態影像所取代，這將造成 CPU 額外的負擔，因此進入此狀態時系統的繪圖速度約比傳統每次重畫的方式提昇 1 到 2 個視框（Frame），我們也藉由系統經常回到 D 狀態的優勢（約 59FPS），使動畫場景的平均速率大於 11FPS、靜態場景的平均速率大於 20FPS，來提昇系統對輸出入裝置之反應速度，以符合電腦遊戲對速度的要求。

然而另一項的負擔是在我們的場景模擬毛筆水墨筆觸後，因爲模擬的筆觸範圍比 OpenGL 繪圖的線條寬度還寬，以致回存的深度值無法含蓋模型邊緣外的筆觸如圖 24 所示，爲保留原有的毛筆水墨筆觸效果則需額外的負擔，因此造成動畫狀態機的 C 狀態

繪圖速度與傳統每次重畫的繪圖速度相當，這是我們當初始料未及的，但此動畫繪圖狀態機仍可應用於 PR 的複雜場景繪圖上。



（a）模型邊緣外的筆觸被覆蓋

圖 24（b）保留模型邊緣外的筆觸

五、 電腦象棋

5.1 簡介

象棋是中國流傳已久的智力比賽遊戲，端賴下棋者所能思考的步數程度，與實際經驗方能贏得比賽，電腦人工智慧（Artificial Intelligence）[9]發展以來電腦下棋此一研究領域一直是印證人工智慧理論與應用的項目[37]。一般來說，在製作電腦象棋程式時通常會劃分成開局、中盤、殘局三個階段，其演算法也會有所不同，開局階段可以資料庫儲存開局棋譜，下棋時就由資料庫查詢得知下一步棋要怎麼走，當然啦不可能儲存無限多個棋譜，因此當局勢發展超出資料庫所存範圍後，就進入了中盤階段。

如果要求的棋力不高，或是初次製作電腦象棋，可將中盤和殘局合併，使用同樣的演算法，一般來說程式中一定會有個評估函數和一個遊戲樹（是一個 Min-Max Tree），爲電腦象棋的核心所在，棋力的高低決定於此，然而評估函數會搜尋遊戲樹，並計算出估計值以決定下一步棋的走法，當進入殘局階段，因爲象棋的殘局有許多特例，通常電腦象棋也會特別爲這些特例建立一個殘局資料庫和特殊的演算法，在實作上我們採用共創軟體聯盟 CChessUG 所開發的的電腦象棋程式，因爲他們的程式模組很容易了解與應用，只需稍加修改即可符合我們的系統需求。

5.2 人工智慧演算法

我們知道電腦象棋之所以會思考主要是依賴核心的評估函數，而評估函數主要的功用在於評估盤面的敵我情勢，透過評估值可以判斷目前的局勢消長，以決定下一步棋的走法，其中影響評估函數的估計結果，主要有下列三項：

1．棋子的價值

棋子的價值是評估局勢最簡單的方法，高手讓對手雙馬或一車來平衡雙方的棋力，就是最好的說明，我們可以看看前人怎麼來設計，應該特別注意的是，象棋的規則勝負在於看誰先吃掉對方的主帥，所以將(帥)的重要性最大但他的價值並不高，我們引用的人工智慧程式其棋子的價值如下所示：

```
//                    將  士  象  馬  車  炮  兵
const base[7]    =   {300 ,300 ,300 ,500 , 850 ,500 ,300};   //平均價值
const range[7]   =   {  0 ,  0,  0,  20,  10,  0  , 40};   //價值的變動範圍
const int BV1[7] = {    //計算基本價值
  base[0]-base[0]*range[0]/100,
  base[1]-base[1]*range[1]/100,
  base[3]-base[2]*range[2]/100,
  base[3]-base[3]*range[3]/100,
  base[4]-base[4]*range[4]/100,
  base[5]-base[5]*range[5]/100,
  base[6]-base[6]*range[6]/100   };
```

2．棋子的棋盤位置價值

在象棋的開局裡，比賽雙方通常會儘速的將車馬包移到重要位置上，尤其是車一定要搶住重要的線，可見棋盤位置對局勢的影響。在爲位置評分時，可以爲每個不同的棋子設計一個二維陣列，用以記錄棋盤上每一個位置的重要性，下面是兵（卒）在棋盤位置的價值設定例子。

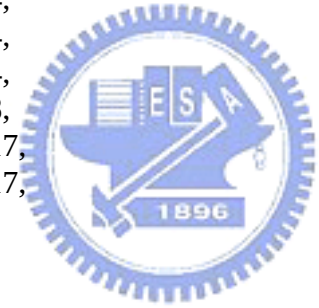
```
//兵卒在不同位置的價值，數位越大價值越高
const int ManBPlus[2][12][11]={
        {{  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
         {  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
         {  0,  1,  2,  3,  4,  4,  4,  3,  2,  1,  0},
         {  0,  1,  2,  3,  4,  4,  4,  3,  2,  1,  0},
         {  0,  1,  2,  3,  3,  3,  3,  3,  2,  1,  0},
         {  0,  1,  1,  1,  1,  1,  1,  1,  1,  1,  0},
         {  0,  0,  0,  1,  0,  0,  0,  1,  0,  0,  0},
         {  0,  0,  0,  0,  0,  2,  0,  0,  0,  0,  0},
         {  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
         {  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
         {  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
         {  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0}},

        {{  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
         {  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
         {  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
         {  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
         {  0,  0,  0,  0,  0,  2,  0,  0,  0,  0,  0},
         {  0,  0,  0,  1,  0,  0,  0,  1,  0,  0,  0},
         {  0,  1,  1,  1,  1,  1,  1,  1,  1,  1,  0},
         {  0,  1,  2,  3,  3,  3,  3,  3,  2,  1,  0},
         {  0,  1,  2,  3,  4,  4,  4,  3,  2,  1,  0},
         {  0,  1,  2,  3,  4,  4,  4,  3,  2,  1,  0},
         {  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0},
         {  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0}}
};
```

3．棋子在棋盤位置上的靈活度

在棋盤裡，每一種棋子都有它自己的走法與可走的範圍，如果在下棋的過程中，讓棋子的可走範圍限制變很多，將會使棋子的平均價值降低，所以在評估局勢時，棋子的靈活度也是考慮的因素，下面是棋子在棋盤位置上的靈活度例子。

```
//                        將  士   象  馬  車   炮  兵
const base[7]=  {300,300,300,500, 850,500,300};  //平均價值
const range[7]= {0   ,   0,   0, 20,   10,   0, 40};  //價值的變動範圍
const int BV2[7]=//計算靈活度
{
 2*base[0]*range[0]/100/4,
 2*base[1]*range[1]/100/4,
 2*base[2]*range[2]/100/4,
 2*base[3]*range[3]/100/8,
 2*base[4]*range[4]/100/17,
 2*base[5]*range[5]/100/17,
 0,
};
```



前述是影響評估函數估計結果的三個主要因素，接下來我們說明遊戲樹與搜尋方法。下象棋時，棋力越高的人，就能看到越遠的步數，一個能看到十三步以後的棋手，比一個只能看到五步棋手的棋力肯定要高。在電腦象棋中可藉由建立遊戲樹來達到往前看的能力。如圖 25 所示。

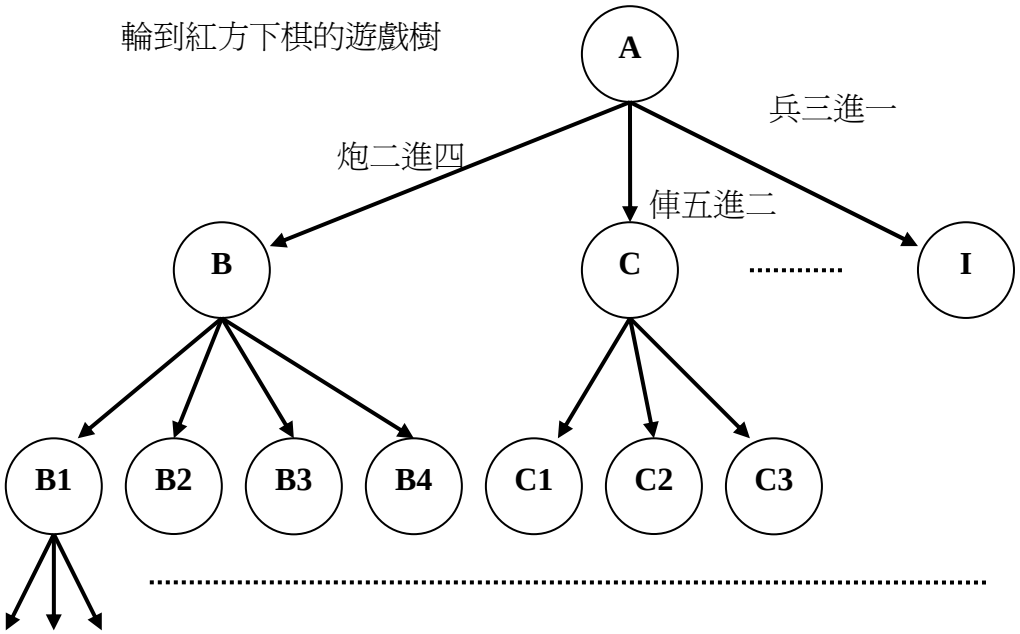


圖 25 遊戲樹範例

當建立好遊戲樹之後，可用廣度搜尋或深度搜尋最佳的走法，爲使搜尋的速度加快，通常會以廣度搜尋先搜尋到某個深度之後，再針對幾個重要走法做深度搜尋。要特別注意的是，評估函數設計得越精準棋力固然越高，但是越精準的評估函數需要越久的計算時間，因爲評估函數對遊戲樹的每一個結點都會以遞迴的方式呼叫自己一次，如果評估函數計算太久，在有限時間內，遊戲樹就不能建立太多層，這將影響到電腦的棋力，因此如何取得一個平衡，就需靠經驗了。

因爲象棋的變化很多，如果對所有可能的走法都要建立遊戲樹，那麼遊戲樹將會變得非常龐大，所以需利用修減演算法來限制遊戲樹的大小，目前配合 Min-Max Tree 最常用的當屬 Alpha-Beta 修減法[9]，我們使用的也是此種方法。

六、 實作結果

本章將呈現實作的結果，我們以 Fly3d v.102 三維電腦遊戲引擎作爲系統開發平台，並以 C++物件導向語言及 OpenGL 實作出完整的三維電腦象棋程式，在設備爲 P4-3.2G ，RAM 512MB 的 PC 平台上，測試我們的結果，測試的影像大小爲 800x600，實作結果說明如下。

圖 26 是 PR Silhouette 的場景執行速率約 29 FPS，圖 27 是柳葉描技法模擬左邊範例、右邊模擬結果，圖 28 是減筆描技法模擬左邊範例、右邊模擬結果，圖 29 是竹葉描技法模擬左邊範例、右邊模擬結果，圖 30 是琴弦描技法模擬左邊範例、右邊模擬結果，圖 31 是棗核描技法模擬左邊範例、右邊模擬結果，圖 32 是綜合技法模擬，表 7 爲各種繪圖狀態之執行速率表列，表 8 表列爲我使用模型的頂點數量、三角面數量及場景物件總數量、頂點總數量與三角面總數量。

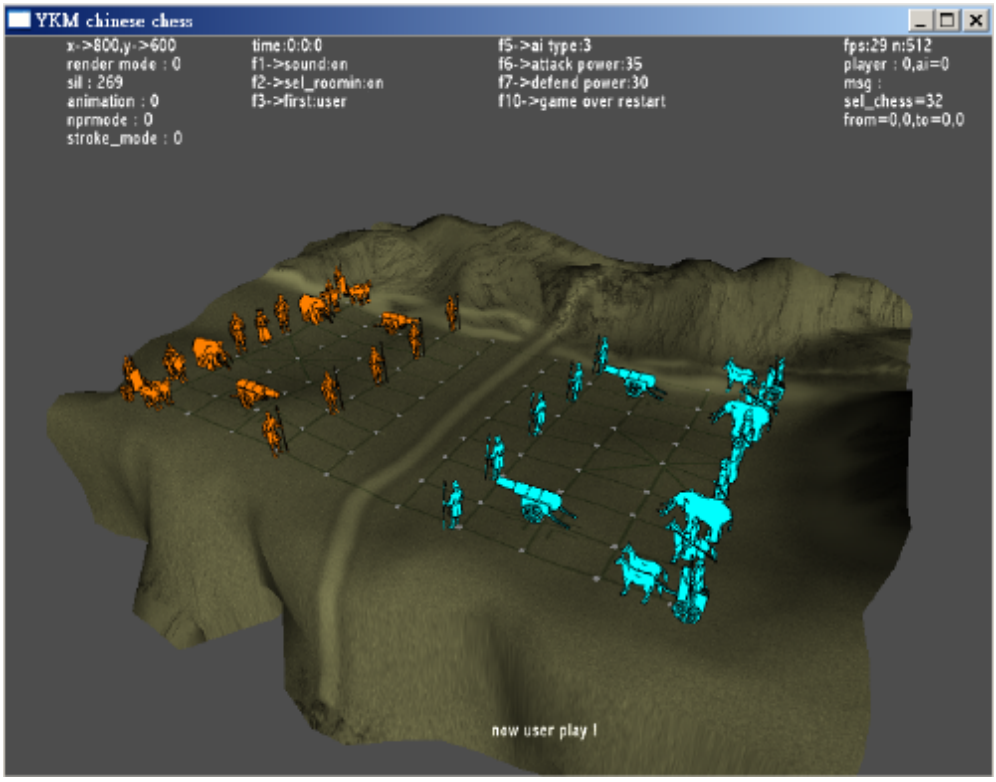


圖 26　PR Silhouette 場景

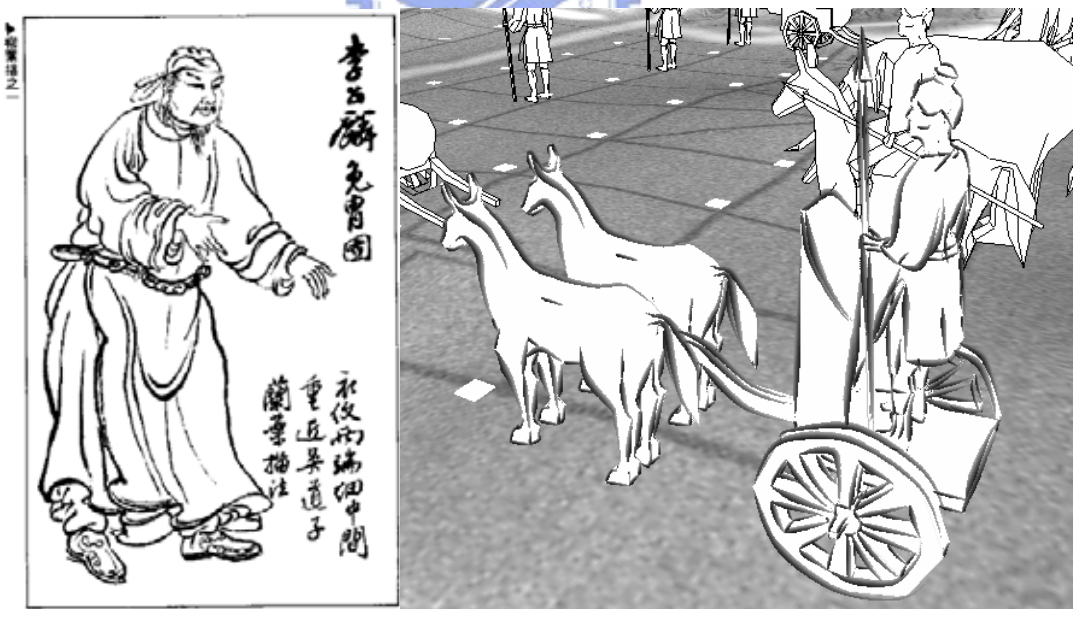


圖 27 柳葉描模擬（stroke_mode = 0）

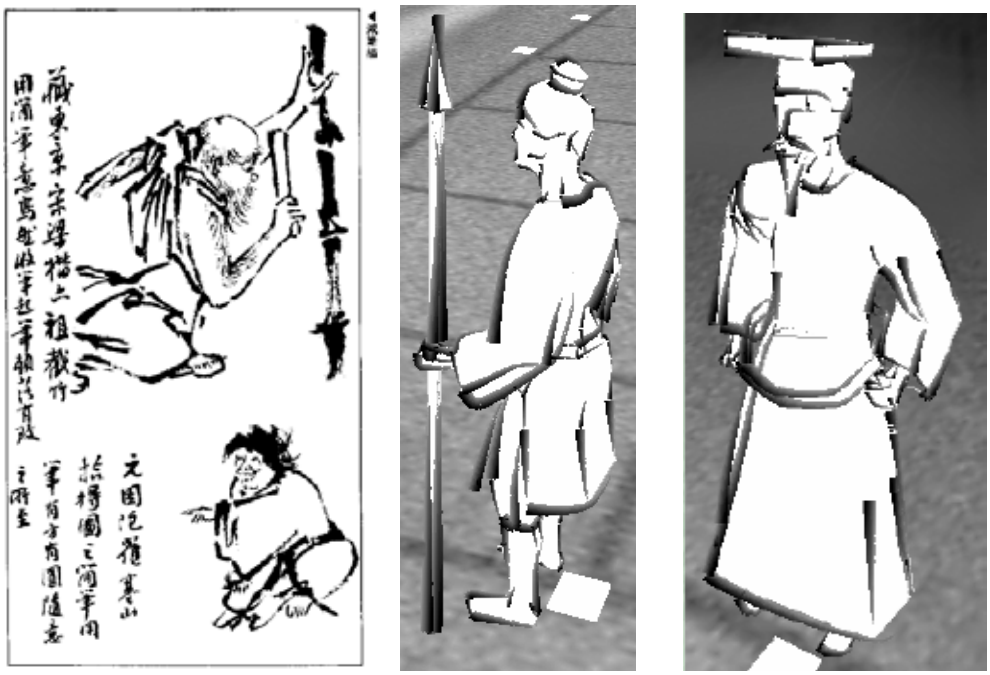


圖 28 減筆描模擬（stroke_mode = 1）

圖 29 竹葉描模擬（stroke_mode = 2）

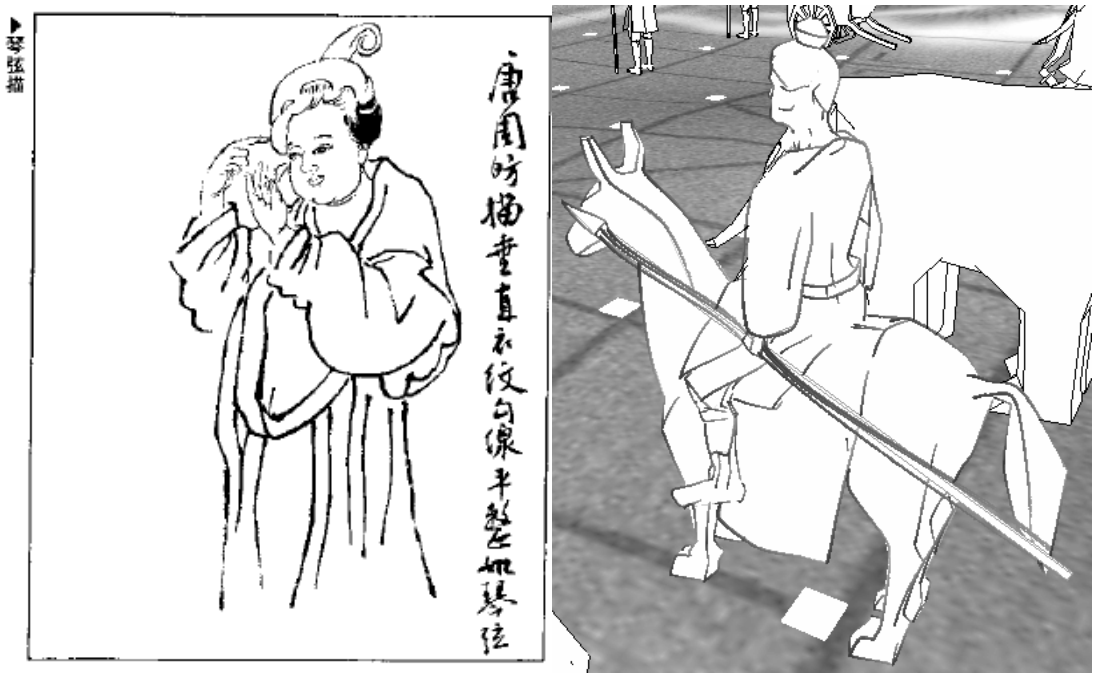


圖 30 琴弦描模擬（stroke_mode = 3）

圖 31　棗核描模擬（stroke_mode = 4）

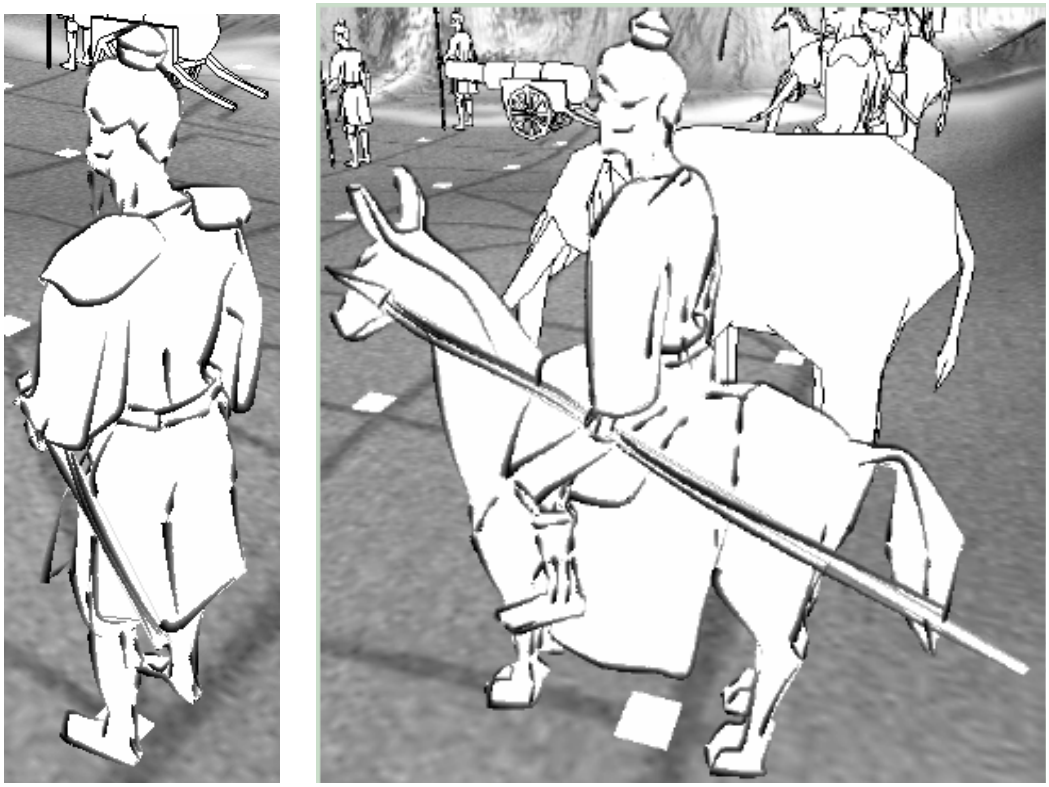


圖 32　綜合技法模擬（stroke_mode = 5）

表 7　執行速率表列

影像大小 800 x 600	State A	State B	State C	State D	平均速率
PR Silhouette 場景	29 FPS	無	無	無	無
NPR 靜態場景	1~4 FPS	無	無	59 FPS	>20FPS
NPR 動態場景	1~4 FPS	1~4 FPS	1~4 FPS	59 FPS	>11FPS

表 8　場景與模型的三角面與頂點數量

	頂點數量	三角面數量
將	429	729
士	519	862
象	290	572
車	1389	2426
馬	840	1445
包	413	672
卒 （A：動畫模型）	505 A：1028	875 A:1606
場景總計：		
物件數量	127	
頂點總數	21604 （A：24219）	
三角面總數	37388 （A：41043）	

七、 結論

本篇論文中，我們參考 Weng [35]的水墨模組，將毛筆模組的模擬機制應用於三維電腦遊戲引擎中，以電腦自動找出輪廓線與特徵線的方法，並綜合 Northup 等人[20,22]提出的方法來繪製 ID 參考影像及處理場景中物體遮蔽問題，最後將可見的線段連接成較長的線段，產生 ***Cardinal Spline*** 的毛筆移動軌跡，以呈現中國水墨人物白描技法的效果，我們根據線條的變化與否分別模擬了五種的技法分別是：柳葉描、減筆描、竹葉描、琴弦描、棗核描及最後的綜合模擬。我們知道畫家作畫時是隨性的寫意，然而在電腦繪圖的領域仍然受到許多的限制，比如說模型的精細度、傳統的繪圖機制限制以及需遵循規律的邏輯運算等等，都影響著結果。

在電腦繪圖上爲使繪圖速度加快，通常會以降低場景的三角面總數量爲考量因素之一，因爲過多的三角面數量會增加 CPU 的計算量而降低繪圖的速度，尤其在 NPR 的繪圖上。在繪圖的速度上我們提出了繪圖狀態機，將傳統的繪圖機制拆成四個狀態分別是：全部繪製（已儲存影像）、先繪靜態部份（並儲存影像與深度值）再繪動畫物件、回存靜態部份深度值再繪動畫物件與將儲存影像直接輸出至螢幕等四個狀態，系統再依照目前所處的狀態及其變化情形，以邏輯推算選擇下一個繪圖狀態來繪圖，用於提昇三維電腦遊戲引擎對輸出入裝置之反應速度，使畫面顯示與遊戲之操作更符合電腦遊戲軟體的需求。在 4.3 節中我們談到，在我們模擬水墨風格的例子上，對動態狀態機中的 C

狀態的繪圖速度幫助不大，但仍可藉由系統經常會到 D 狀態的優勢來提升系統的整體速度，使動態場景平均速率大於 11FPS、靜態場景平均速率大於 20FPS，以符合速度的要求。而我們提出的方法亦可應用於複雜的 PR 場景繪圖中。

未來我們可參考 Yeh[36]提出的方法將白描的模型內部加以上色，使之呈現更豐富的色彩效果，對碰撞部分加以處理使顯示完整的棋子移動過程，在速度方面找尋其他加速的方法譬如 LOD 與 Pre-process 的方法，使 NPR 水墨的效果更能夠應用於三維電腦遊戲引擎及虛擬實境中。

參考文獻

1. 王耀庭編，「山水畫法1、2、3」，雄獅圖書公司，民國七十三年三月。

2. 鄭明編著，「中國山水畫技法」，藝風堂出版社，民國七十六年三月出版。

3. A. Appel, “The Notion of Quantitative Invisibility and the Machine Rendering of Solids” , Proc. ACM National Conf., Thompson Books, pp. 387-393, 1967.

4. J.W. Buchanan and M.C. Sousa, “The Edge Buffer: A Data Structure for Easy Silhouette Rendering” , Proc.1st Int’l Symp. Non-Photorealistic Animation and Rendering, ACM Press, pp. 39-42, 2000.

5. J. Curtis Cassidy, D. Anderson Sean, E. Seims Joshua, W. Fleischer Kurt, H. Salesin David, “Computer-Generated Watercolor” , Proc. of SIGGRAPH’97, pp. 421-430, 1997.

6. G. Elber and E. Cohen. “Hidden curve removal for free form surfaces” , In Proceedings of SIGGRAPH ’90, pp. 95–104,August 1990.

7. C. EVERITT, “One-pass silhouette rendering with geforce and geforce2.” NVIDIA Corporation White Paper.

8. B. Freudenberg, “Real-Time Stroke Textures” , (Technical Sketch) SIGGRAPH 2001 Conference Abstracts and Applications, p. 252, 2001.

9. F.Luger George, A.Stubblefield William, “Artificial Intelligence : Structures and Strategies for Complex Problem Solving”, Addison Wesley Longman, INC.1999.

10. B. GOOCH, P.-P. J SLOAN, A. GOOCH, P. S. SHIRLEY,, AND R. RIESENFELD, “Interactive technical illustration.” In 1999 ACM Symposium on Interactive 3D Graphics (April 1999), ACM SIGGRAPH, pp. 31–38. ISBN 1-58113-082-1.

11. Horace H S Ip, Helena T F Wong, “Calligraphic Character Synthesis Using a Brush Model,” Proc. of Computer Graphics International 1997, page 13-21, 1997.

12. Chih-Wei Hsu, “The Synthesis of Rock Textures in Chinese Landscape Painting” , Ms thesis, National Chiao Tung University, June 1999.

13. Sheng-Wen Huang, Der-Lor Way, Zen-Chung Shih, “Phisically-Based Model of Ink Diffusion in Chinese Ink Paintings” , Proc. of WSCG 2003.

14. T. Isenberg, N. Halper, and T. Strothotte, “Stylized Silhouettes at Interactive Rates: From Silhouette Edges to Silhouette Strokes” , Computer Graphics Forum (Proc. Eurographics 2002), vol.21, no. 3, pp. 249-258, 2002.

15. D.J., B., AND J.F., H. “Rapid approximate silhouette rendering of implicit surfaces.” In Proceedings of Implicit Surfaces 98 (June 1998), pp. 155–164.

16. D. E. JOHNSON, AND E. COHEN, “Spatialized normal cone hierarchies.” In 2001 ACM Symposium on Interactive 3D Graphics (March 2001), pp. 129–134. ISBN 1-58113-292-1.

17. A.W. KLEIN, W. LI, M.M. KAZHDAN, W. T. CORREA, A. FINKELSTEIN, AND T.A.FUNKHOUSER, “Non-photorealistic Virtual Environments.” In Proceedings of SIGGRAPH 00, Computer Graphics Proceedings, pp.527-534.

18. Gao Ling Mei, “Chinese Painting by Chang Da-Chien,” Published in 1988 by Art Book Co., Ltd.

19. L. Markosian, M. A. Kowalski, S. J. Trychin, L. D. Bourdev, D. Goldstein, and J. F. Hughes, “Real-time non-photorealistic rendering” , Proc. of SIGGRAPH’97, pp. 415–420, 1997.

20. A. Michael Kowalski, J. D. Lee Markosian, Northup, Lubomir Bourdev, Ronen Barzel, S. Loring Holden, and John Hughes, “Art-Based Rendering for Fur, Grass, and Trees,” Proc. of SIGGRAPH’99, pages 433-438, August 1999.

21. JASON MITCHELL. “Real-time image-space outlining for non-photorealistic rendering.” siggraph 2002 session notes. ATI Corporation White Paper.

22. J.D. Northup and L. Markosian, “Artistic Silhouettes: A Hybrid Approach” , Proc. 1st Int’lSymp. Non-Photorealistic Animation and Rendering, J.-D. Fekete and D.H. Salesin, eds., ACM Press, pp. 31-37, 2000.

23. E.PRAUN, H.HOPPE, M.WEBB, A.FINKELSTEIN, “Real-Time Hatching” In Proceedings of SIGGRAPH 01, Computer Graphics Proceedings, pp.581-586, 2001.

24. Ramwsh Rashkar and Michael Cohen , “Image Precision Silhouette Edges” , Symposium on Interactive 3D Graphics , April 26-29 1999.

25. P.Grimaldi Ralph, ”Discrete and Combinatorial mathematics”, Addison Wesley

26. Ramesh Raskar , “Hardware Support for Non-photorealistic Rendering” , In proceedings of siggraph / eurographics graphics hardware workshop , August 2001.

27. M. P. SALISBURY, S. E. ANDERSON, R. BARZEL, AND D. H.SALESIN, “Interactive pen-and-ink illustration.” In Proceedings of SIGGRAPH 94, Computer Graphics Proceedings, pp.101-108, 1994.

28. M. P. SALISBURY, M.T. WONG, J. F. HUGHES, AND D.H SALESIN, “Orientable textures for image-based pen-and-ink illustration.” In Proceedings of SIGGRAPH 97, Computer Graphics Proceedings, pp.401-406, 1997.

29. M. SALISBURY, C. ANDERSON, D. LISCHINSKI, AND D. H SALESIN, “Scale-dependent reproduction of pen-and-ink illustrations.” In Proceedings of SIGGRAPH 96, Computer Graphics Proceedings, pp.461-468, 1996.

30. M. C. Sousa, and J. W. Buchanan, “Observational model of blenders and erasers in computer-generated pencil rendering,” Proc. of Graphics Interface’99, pp. 157–166, 1999.

31. M.C. Sousa, and J. W. Buchanan, “Computer-generated graphite pencil rendering of 3d polygonal models,” Computer Graphics Forum, Vol.18, No.3, pp. 195–208, 1999.

32. Steve Strassmann, ”Hairy Brushes,” Proc. of ACM SIGGRAPH’86, pp. 225-232.

33. Der-Lor Way, Zen-Chung Shih, “The Synthesis of Rock Textures in Chinese Landscape Painting”, Computer Graphics Forum, Vol.20, No.3, pp. C123-C131, 2001.

34. Der-Lor Way, Yu-Ru Lin, Zen-Chung Shih, “The Synthesis of Trees in Chinese Landscape Painting Using Silhouette and Texture Strokes”, Journal of WSCG, Volume 10, Number3, pp. 499-507, 2002.

35. Shan-Zan Weng, Zen-Chung Shih, Hsin-Yi Chiu, “The synthesis of Chinese Ink Painting” , National Computing Symposium’99, page 461-468, 1999.

36. Jun-Wei Yeh and Ming Ouhyoung , “Nor-Photorealistic Rendering in Chinese Painting of Animals” , pp.1220-1224 and pp.1262, Journal of System Simulation (ISSN 1004-731X), Vol.14, No.6 ,2002

37. Shi-Jim Yen, Jr-Chang Chen, Ting-Ning Yang, Shun-Chin Hsu, “Computer Chinese Chess,” ICGA Journal, Vol. 27, No.1, March 2004, pp. 3-18, ISBN 1389-6911.

38. Fly3d Game Engine Software http://fly3d.com.br

39. NPRQuake Software http://www.cs.wisc.edu/graphics/Gallery/NPRQuake/