國立暨南國際大學資訊工程學系

碩士論文

指導教授：陳履恆博士

建構在 3D Skeleton 架構上的 LOD 演算法 Continuous Level of Detail Based on 3D Skeleton Structure

碩士生：包明樵

中華民國九十二年六月二十六日

致謝

轉眼間我已在暨南大學度過了研究所的學習生涯，回想著暨大的一切，心中全是學習的甘甜與難捨。

首先，我要感謝我的指導教授陳履恆老師，在課業上與日常生活中給予我非常多的指導，除了討論研究領域的相關議題外，老師還時常教我們許多待人處世的道理，其中最令我難忘的是他誠懇謙虛的處世態度與嚴謹的研究方法，我相信老師所教授的一切一定令我受用無窮。

也謝謝我親愛父母，在我求學的生涯中一直默默的支持，使我能全心全力的在課業上而無後顧之憂。

最後，我要感謝親愛的同學以及學弟，有你們的陪伴，才能讓我這一路順利不少。

摘要

Level of Detail(LOD) 是藉由物件跟觀測者的相對距離來改變三維模型的解析度，目的是爲了在執行繪圖時能夠減少計算的時間。開發一個，在不破壞物體形狀的前提下，有效率的建立 3D 物體 LOD 的手法。是目前在 CG 研究領域中，極具應用價值的研究課題，例如：醫學工程、遊戲引擎設計等等。在本論文中，我們提出了一個架構在三維骨架 (3D skeleton) 上的 LOD 演算法。不同於一般的手法的是 : 我們的演算法是先從決定代表物件基本形狀的頂點開始。

在計算物件的 3D skeleton 過程中，物件的頂點將分爲兩個子集合， G 和 M。G 集合用來求取 skeleton 節點的頂點集合。因此，與 skeleton 有直接關聯的 G 中的頂點，將被給予較高的權重。 G 的點數視 3D skeleton 中所使用的 Length of Path 而定。一般而言，我們使用模型平均邊長的兩倍作爲 Length of Path 來建立 skeleton。這表示我們先從模型的 n 個頂點中挑出 n/2 個 " 較重要 " 的頂點 (G)。另一方面，Length of Path 也決定了 3D skeleton 的 level 數目。 G 將會被分類爲 k 個子集合。

繪圖時，利用 3D skeleton 的結果動態的計算物件的 LOD。 LOD 的計算是依據 skeleton section 與 camera 之間的相對方位。此外，也考量到一些其他的因素，包括：local illumination、 screen-space projection、visibility culling 與 silhouette boundaries.

藉由導入 3D skeleton 的架構，我們的系統在執行模型簡化時便能夠有效率的保存物件的外型。在本論文中，我們成功的開發出一個 real-time、continuous、shape preserving 與 view-oriented 的 LOD 演算法。

關鍵字：電腦圖學、LOD、3D skeleton、虛擬實境

Abstract

The main purpose of Level of Detail (LOD) is to reduce the computing cost at the rendering stage by switching among the multi-resolution 3D models according to the distance between the object and viewer. Developing an efficient continuous LOD method, which preserves the silhouette of object at low resolution, is one of the important research topics of Computer Graphics. LOD is also a very practical issue to many fields such as medical engineering, game engine design, etc. In this paper, we proposed a new LOD algorithm based on the 3D-skeleton structure of objects. Instead of starting from picking up vertices to be eliminated as most of LOD algorithms do, our method first determines the minimum vertices set which represents the basic geometric shape of object.

During the extraction of the 3D skeleton structure of objects, the vertices set of the original model is divided into two subset, G and M. Vertices belongs to G set are used to construct the skeleton nodes. Since we consider the skeleton as a kind of extremely simplified geometric 3d shape, it is very intuitive and reasonable to assign higher weighting value to the vertices belongs to G. The ratio of G's size to M's size depends on the length of path used in the 3d skeleton extraction algorithm. Generally speaking, the length twice as long as the average edge-length of the 3d model creates a satisfactory result. In another word, half of the vertices are more important than the rest to us in this case. The length of path used also determines the number of levels of 3d skeleton. Thus the L set is further clustered into k sub-sets.

During the rendering stage, our algorithm dynamically computes the continuous LOD of object based on the previous analysis results. The LOD is then computed according to the relative position and orientation between each skeleton section and camera. Besides, our algorithm also considers other factors, includes local illumination, screen-space projection, visibility culling and silhouette boundaries.

By introducing the 3D skeleton structure, our system can perform the model simplification with the shape preserved in an efficacious and intuitive way. In this paper, we successfully developed a real-time, continuous, shape preserving, and view-oriented LOD algorithm.

Keywords: Computer Graphics, LOD, 3d Skeleton, Virtual Reality

目錄

圖例目錄

1 Introduction

在電腦圖學的領域中，Levels of Details(LOD) 演算法是一個極具應用價值的研究題材。LOD 的主要目的是在於降低的模型複雜度，但是卻又能同時保有高度的繪圖品質，進而達到加快繪圖速度，以節省繪圖時間。尤其在一些需要大量模型或是即時繪圖的工作時，例如飛行模擬，LOD 演算法更是扮演重要的角色。它可以提供繪圖引擎更高的效率，進而能夠處理更多更大量的資料。一個好的 LOD 演算法應該具備以下幾點條件： 1. 要能盡量維持物體形狀。 2. 要能大量的降低模型複雜度。 3. 要能夠即時的運算。

本論文的主要目的在於開發一個在不破壞物體形狀的前提下，即時 (Real-Time) 且有效率的建立 3D 物體 LOD 的手法。而其中的關鍵之處，就是我們導入了三維骨架 (3D skeleton) 的架構。3D skeleton 和模型本身可以說是一體兩面的資訊。skeleton 代表了模型的基本形狀，也就是大致上的外觀；而模型則是表現了較細微的部分。我們希望能夠結合 3D skeleton 與 model 本身的資訊再導入 LOD 系統中，進而能夠求出在較低解析度下時，模型能夠使用少量的頂點但是卻又能保持物件的外觀形狀。

本系統在實作上分成三個部份，第一個部份是 3D skeleton 的計算，我們希望透過直接計算就能找出物件的 skeleton 資訊。第二部分是刪除點的選擇，也就是模型的簡化步驟 (Model Simplification)，我們將整合過去 LOD 系統中常使用的判斷方式與觀念，再輔以 skeleton 的資訊，以找出最適合的刪除點並確保物件的基本形狀。第三部份是三角形的重建 (Dynamic Triangulation)，由於 LOD 的系統需要即

時的運算，所以我們運用一些方式來達成，而其中，skeleton 的資訊對於三角形的重建也大有幫助。

1.1 文獻探討

在 3D skeleton 方面的技術有很多，例如，使用 Voronoi graph and Delaunay tessellation[2]， generalized potential field[6]，visible repulsive force[8]， radial basis functions[7]。Skeleton 對物件來說是一種相當好的外型描述，它提供物件最基本的外觀資訊，若能再輔以一些 geometry 的資訊，對於重建物件的外型 [3][4][5]，將有莫大的幫助。在本論文中我們將參考 [1] 的做法，因爲他不僅僅能提供 skeleton 本身的資料，而更重要的是要他也能提供我們 skeleton 和 surface model 之間的相互關係，透過 3D skeleton 的計算讓我們能夠輕易的取得外觀的資訊，這讓我們在執行 model reconstruction 時更爲精確，並能確保基本形狀。我們參考此文獻的做法並加以適當的修改，使其更加廣泛的應用於一般模型資料，也讓輸出的結果能夠提供更爲精確的資料，更符合我們 LOD 的需求。

在 LOD 方面，早期的做法利用 Delaunay triangulation[9][10] 或 constrained delaunay triangulation[13][14] 來重建三角形是一個不錯的做法，這種能夠提供不錯的速度，且所重建的三角形也能盡量維持較好的形狀。但是，這樣的方式不適合一般的模型來使用，因爲 Delaunay triangulation 在處理有凹陷的部分有可能會出現錯誤，所以這樣的做法是較適合應用於地表模型的處理；也有利用 regular grid 的技術 [11][12]，這也可以達到 real-time 的運算，但是由於資料結構的限制，並不適用一般模型。

H. Hoppe 提出的 progressive mesh[15][16][17][18] 的做法，利用預先計算好的衰退紀錄，在執行 triangulation 的時候就可以大幅減少計算時間，這給予我們在做 triangulation 時給予很大的啓示，雖然與我們的做法不同，但是在基本觀念上亦有異曲同工之妙。此外，文獻 [19] 中提出了相當多 Model Simplification 的觀念，其中的觀念包含了 :local illumination，screen-space projection，visibility culling，silhouette boundaries 可說是集合了近年來 LOD 的做法與技術，在本論文也應用很多其中的觀念與想法。

1.2 研究目的

在 LOD 系統中引入 3D skeleton structure 的資料可說是一項嶄新的應用，skeleton 提供了對物件外型一種很好的描述。我們將在接下來的論文中慢慢印證這個想法，並以實驗結果來證明 3D skeleton 是非常有用的資訊之於 LOD system。而本論文的最主要目的在於結合 3D skeleton，開發一個 : 即時的演算速度，不破壞物體的基本形狀，Continuous 和 View-oriented 建立 3D 物體 LOD 的手法。

1.3 論文編排方式

本論文的編排方式如下：在第二章節，將詳細介紹我們所提出的 LOD 演算法 (A new LOD method based on the 3D skeleton structure)，其中包含三個小節，我們會先介紹 3D skeleton 演算法的過程，接著再詳述 LOD 的 "Model Simplification" 和 "Dynamic Triangulation" 的部分；第三章節有實驗的成果與討論；第四章節我們會詳加探討本論文的得失以及尚未達成的部分，並討論未來可能的發展。

2 Algorithm: A new LOD method based on the 3D skeleton structure

本論文中將提出一個新的建構在 3D skeleton 架構上的 LOD 演算法。此演算法分爲 2 部分，首先，我們要預先對模型做 3D skeleton 的計算，並求出 skeleton 與 model 的相互關係，透過這個動作，我們將可以把模型分成兩部分：與 skeleton 對應的部分和其餘的部分，與 skeleton 對應的部分可以用來代表模型的基本形狀，所以在 Model Simplification 時將會給予較高的權重，使其較不容易被刪除，因此可以比較容易維持模型的外型；然後在接下來的在 Model Simplification 的步驟時，我們將參考：skeleton，screen projection，illumination，以及 view 等等因素來決定刪除點，由於我們加入 skeleton 的考量，所以在 lower resolution 時仍然可以維持住物件的外型不致於被破壞。

2.1 3D skeleton

對於任何一個 surface model，我們將參考 [1] 中所使用的方法，其目的是直接透過計算，勾勒出此物體的 skeleton。這方面的研究，對於醫學方面的研究有非常大的幫助，例如人的腸道，血管等等呈圓柱狀的物體，非常適合此種方法來重建其 3D skeleton。

2.1.1 3D skeleton algorithm

關於 3D skeleton 的演算法有很多，但是，以目前這個技術是最適合用於我們的模型結構。我們將使用 paper 所提出的 skeleton algorithm 作爲我們的演算法基礎，並修改其中一些演算法，使其更適合一般的 model.。

由於文獻中所採用的資料結構爲單純的點資料，並非一般常用的 surface model。所以我們在實作上與文獻的做法略有不同，基於 surface model 能提供 edge 的資訊，所以我們不採用文獻的近似方式，改以精確的 edge 當作算最短距離時的路徑。此外，我們也加入了 multi-source 來對 skeleton 作修正，讓取得的 3D skeleton 更爲正確。

對於任何一個 surface model，首先，依循下列四個步驟來完成物體的 skeletal curve：

1. 首先選定一個起始點，稱爲 "SourcePoint"：

   對於這個 skeleton algorithm 的基本原則，source point 的選擇是很重要的，如圖 1和圖 2所示，它將會影響未來取出的 skeleton 的形狀，一般而言，若模型的形狀越趨近於狹長形狀，則計算出來的 skeleton curve 效果越好。在此我們建議 source point 的選擇，以模型中相距最遠的兩個 vertex 的其中之一。如此，將可以保障我們做出來的 skeleton curve 是最能代表模型的主體形狀。

2. 算出最短路徑：

   接著，我們計算出模型中其他 vertex 到 source point 的最短路徑 (shortest path)，我們使用的是 "Dijkstra's Shortest Path" 演算法，並且將結果存在一個陣列

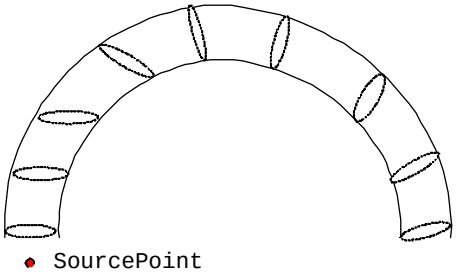


圖 1: 不同 sourcrpoint 對 skeleton 的影響示意圖 .a

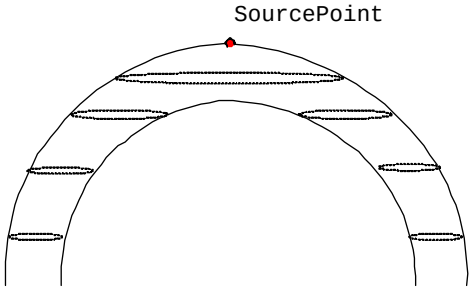


圖 2: 不同 sourcrpoint 對 skeleton 的影響示意圖 .b

中，稱爲 "distance map"。在執行 Dijkstra's shortest path 時，我們使用模型上的邊當作是路徑。因爲我們使用到原本模型的 edge 當作路徑，所以計算出來的最短距離將會是比較正確的。此資訊將會是我們要把模型分成數個 level sets 的主要依據。

3. 設定一個 k 值：

k 值就是我們要把模型劃分成的 level 數。依據模型中其他 vertex 到 source point 的最短距離以及 k 值，將模型分成 k 個 level。如果 k 越大，則做出的 skeleton 越精密，當然做出來的 skeleton curve 將會越好，越接近實際狀況。但是也不能無限制的增加 k 值。k 越大，對於後來的 LOD 計算的負擔的也會增大。而且，有些 skeleton 的資訊將會重疊，這可能會使 LOD 演算法造成困擾，不利

於在做 LOD 的執行速度。因此，我們建議 k 值的大小以下面的計算產生:

$$k = \frac{maxDistance}{avgDistance * 2}$$

其中 ,maxDistance 是模型中相距最遠的兩點的距離，而 avgDistance 是模型的平均邊長。如此得到的 skeleton 用於 LOD 的計算，將可以得到最大的效益。關於 k 值的分析將在第六章節探討。

4. 分支測試：

最後，對每個 level 做分支 (branches) 的測試，看是否有 branches 的存在。若有 branches 則必須將此 level 分開並記錄 branches。然後，取同一個 level 中的所有 vertex 的中點，是爲"skeleton point"，若有 branch 則取同 branch 的 vertex，再將 skeleton point 按照 level 的順序連接就可以得到 skeleton curve。

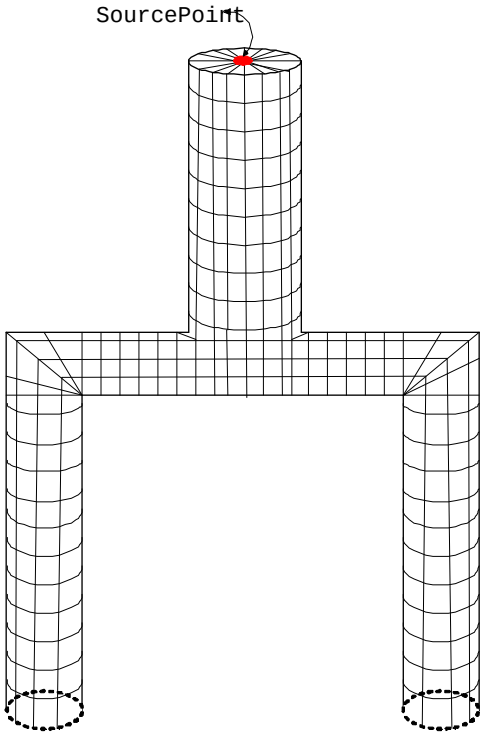


圖 3: 選取一個適當的 SourcePoint。

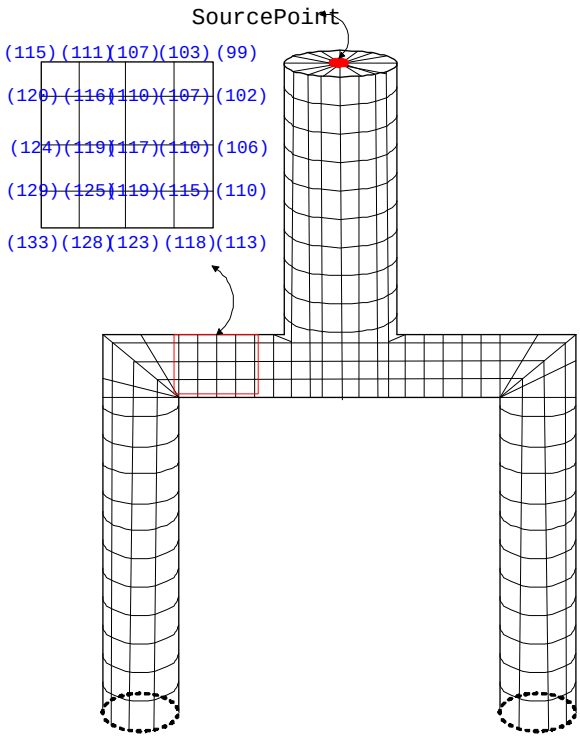


圖 4: 算出所有點到 SourcePoint 的 sjortest path。

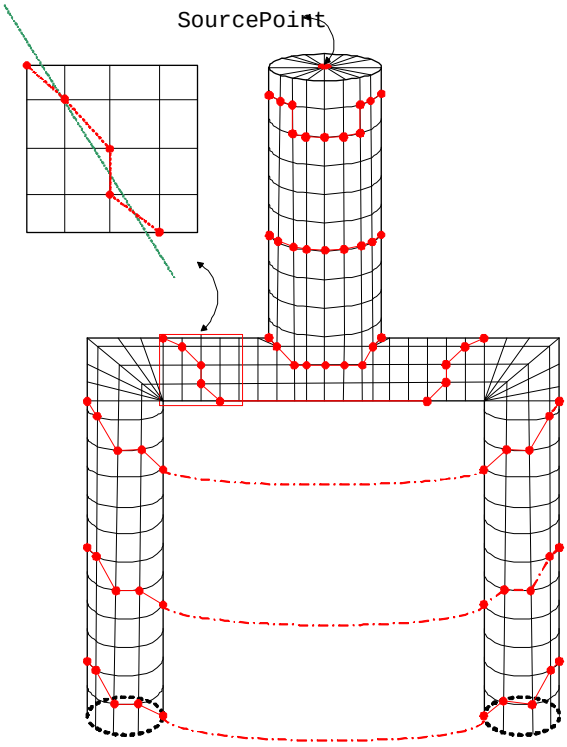


圖 5: 將點分成 k 個 level。

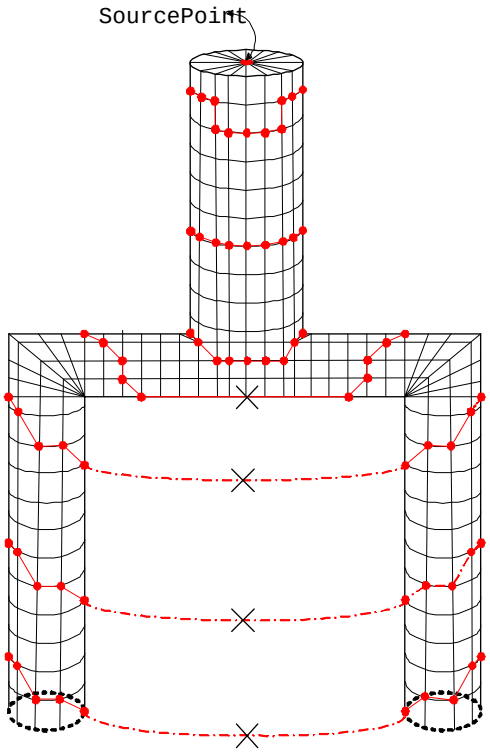


圖 6: 做 branches 的測試。

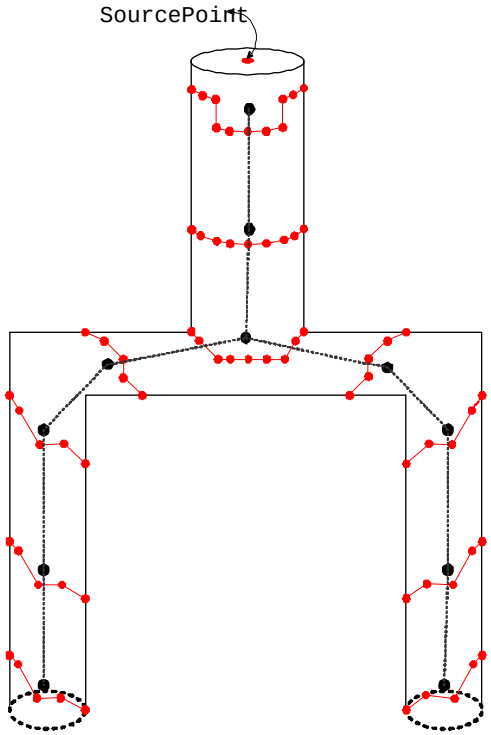


圖 7: 連接中點可得 skeleton。

2.1.2 branches testing

在計算出模型中其他 vertex 到 source point 的最短路徑 (shortest path) 以後，接下來就是依據 k 值以及 distance map 將 model 區分成數個 level sets，所以，每個 level set 都會有一組模型表面的點集合相對應，X = X1, X2, ..., Xn。接著，我們必須將此 level 區分成不同的 subsets，讓 subset 跟模型上的 branches 相符合。我們可以利用 Dijkstra’s shortest path 來求出 X 集合中相鄰的一組點，所謂的相鄰指點與點之間相距不會超過數個 edges。做法如下：1. 首先，先從 X 中任意挑選一點 Xa，以此點當作 source point；2. 算出在 X 中的其他點到 Xa 的 shortest path，並將離 Xa 只相隔 3 個 edges(可自由調整此參數) 的點加入 Xa 的集合； 3. 以”新加入點”當作 source point，重複步驟 2。我們便可以利用像這種簡單的方式就將 X 集合分成合適的 subsets(branchs)。

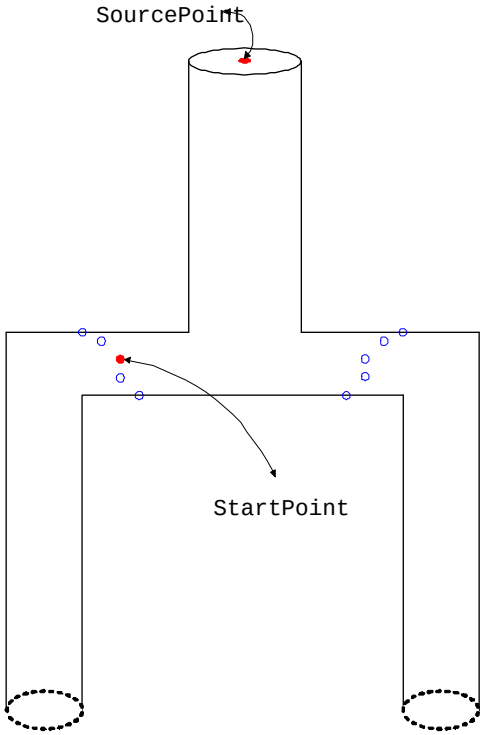


圖 8: 任意選取一個起始點。

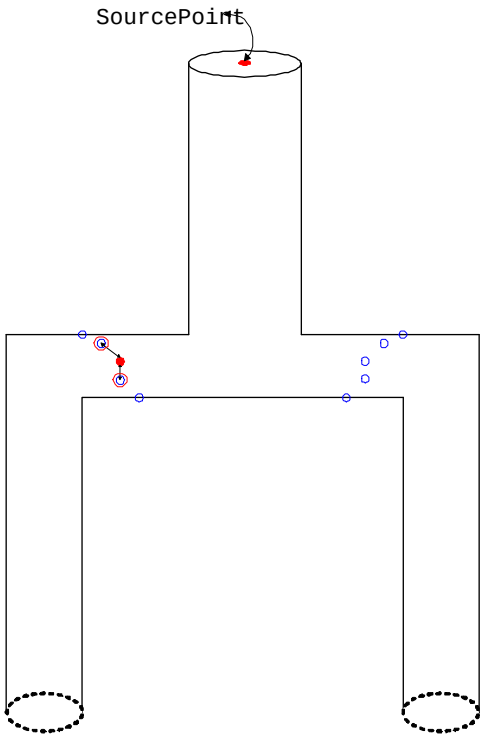


圖 9: 找離此點距離最近的點 (相距 3 個 vertices 內，依據模型調整)。

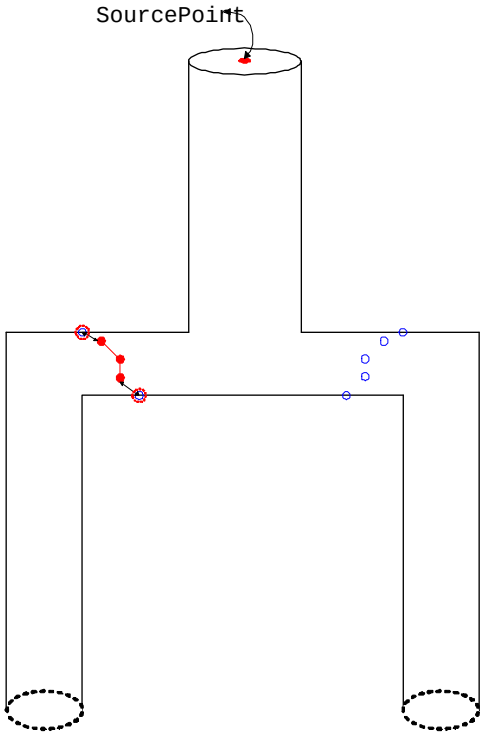


圖 10: 加入新的點並重複步驟 2。

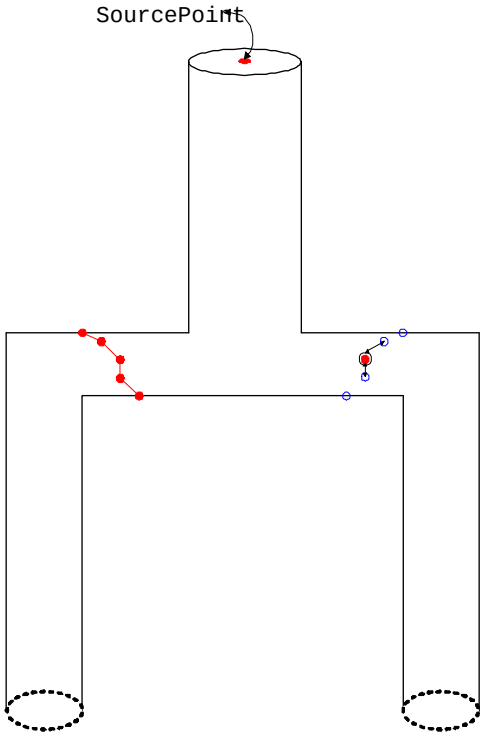


圖 11: 直到沒有新加入的點，再從剩餘的點選取一起始點。

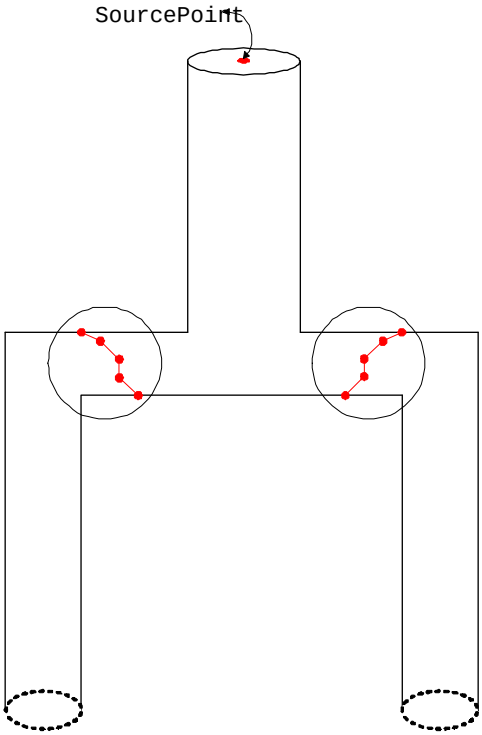


圖 12: 重複步驟 2 4，直到所有點都被使用，就可以求出 branches。

2.1.3 3D skeleton 實驗結果

以下是一些 skeleton 的實驗結果，我們的執行環境爲：AMD Althlon(tm)1.2G，256M RAM，GeForce2 MX100/200，程式使用 visual C++(using MFC) 撰寫。

我們已經能夠成功的獲取到 skeleton 的資料以及 skeleton 和 model 之間的相關資訊，由圖 [13][14][15] 中，我們可以了解到，選擇 source point 對 skeleton 有相當大的影響，如果 source point 選擇的不好將會造成 skeleton 的資料有所偏差。

此外，我們發現到，在有 branches 的地方常常因爲 k 值取的太小，導致容易遺失一些資訊，所以在 branches 的地方會有一些誤差產生，但是 k 值取的太大又會影響執行 LOD 時的效能。所以最好的做法是採用非固定的方式，也就是說在 branch 的地方就減少每個 level 間的距離，而其他地方就使用一般正常的距離。

另一個解決的方法是，採用 multi-source 的方式，也就是說，我們不僅僅採用 one source，而是由多個 source poinr 來決定 skeleton 的形狀。但是這會牽涉到一個問題，就是 skeleton 與 model 間的相關資訊有可能會有重疊或交錯的情況發生，這是使用 multi-source 要注意的問題。圖 [16] 就是利用 multi-source 的方式對 skeleton 做修正，我們可以發現，在魚鰭 (branches) 的地方有明顯的改善。

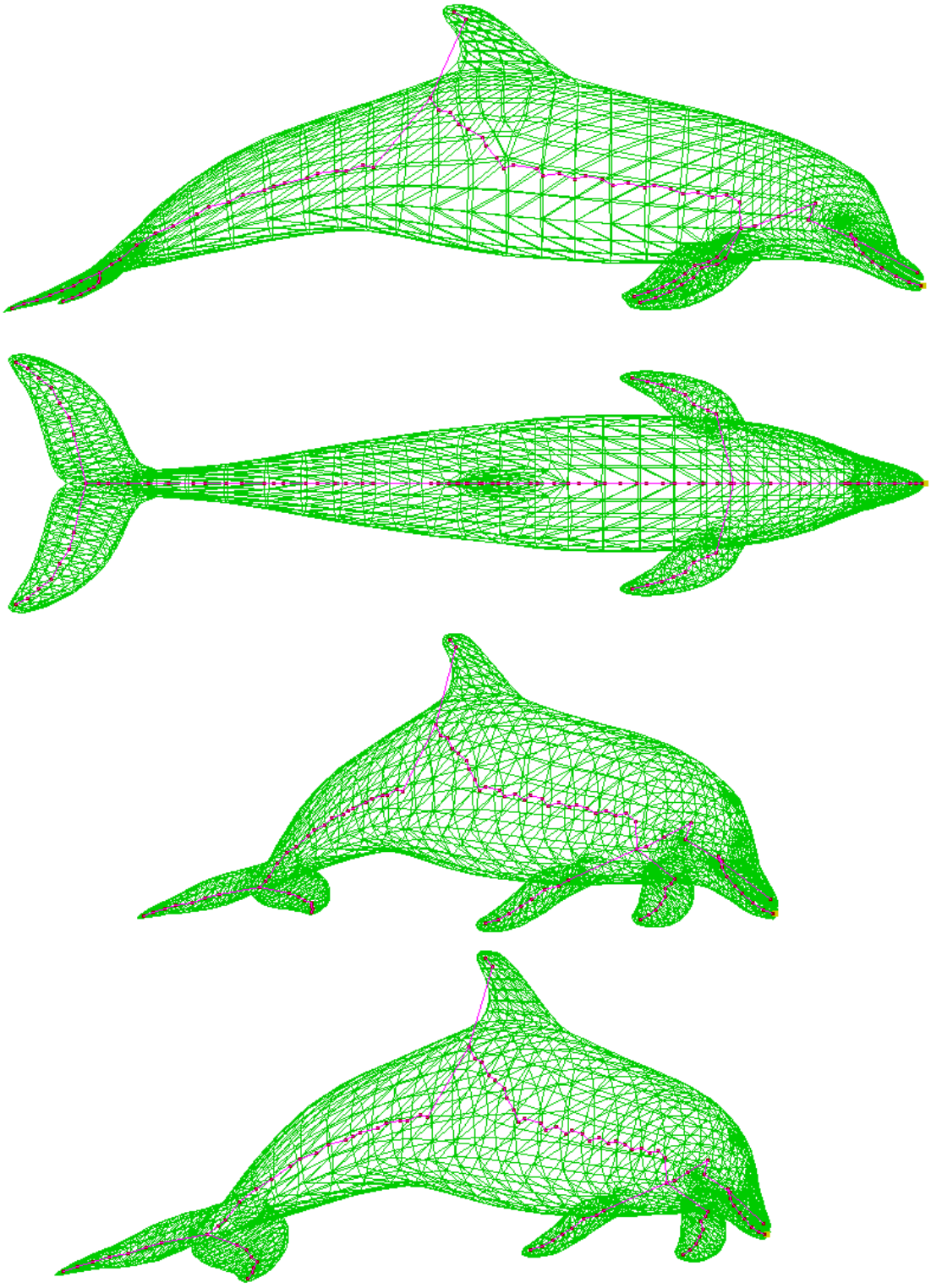


圖 13: SourcePoint 在頭部。

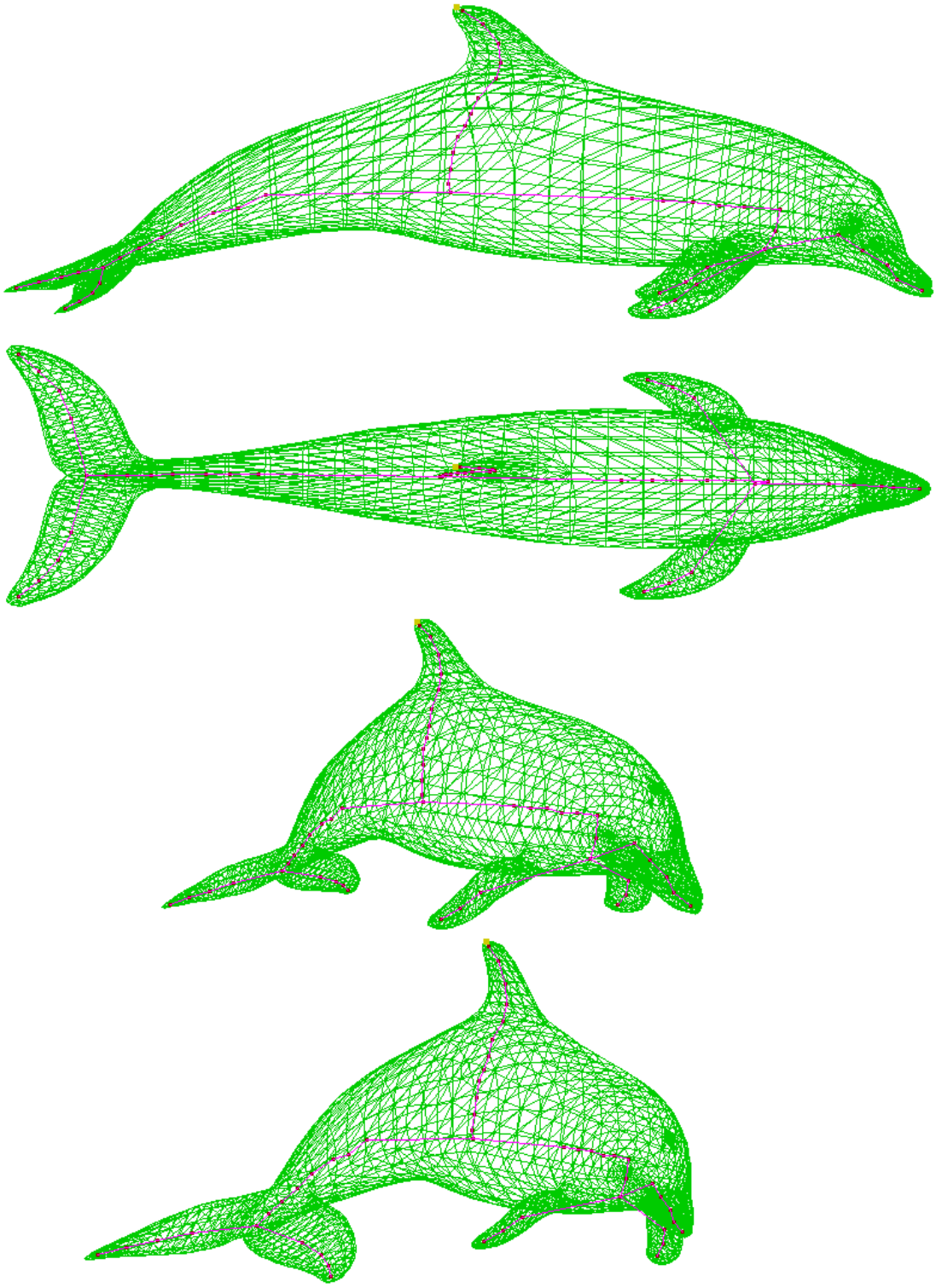


圖 14: SourcePoint 在背鰭。

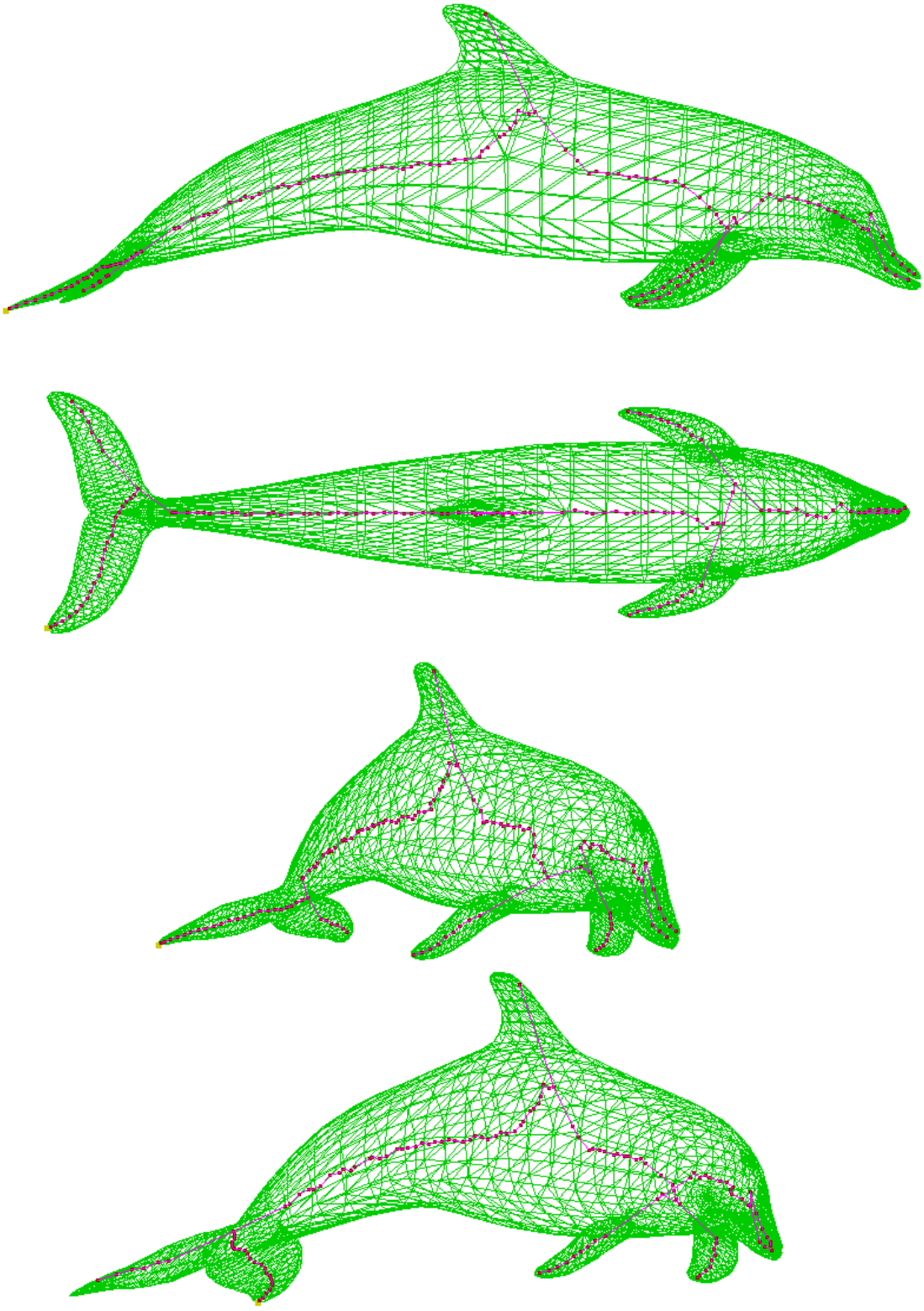


圖 15: SourcePoint 在尾鰭。

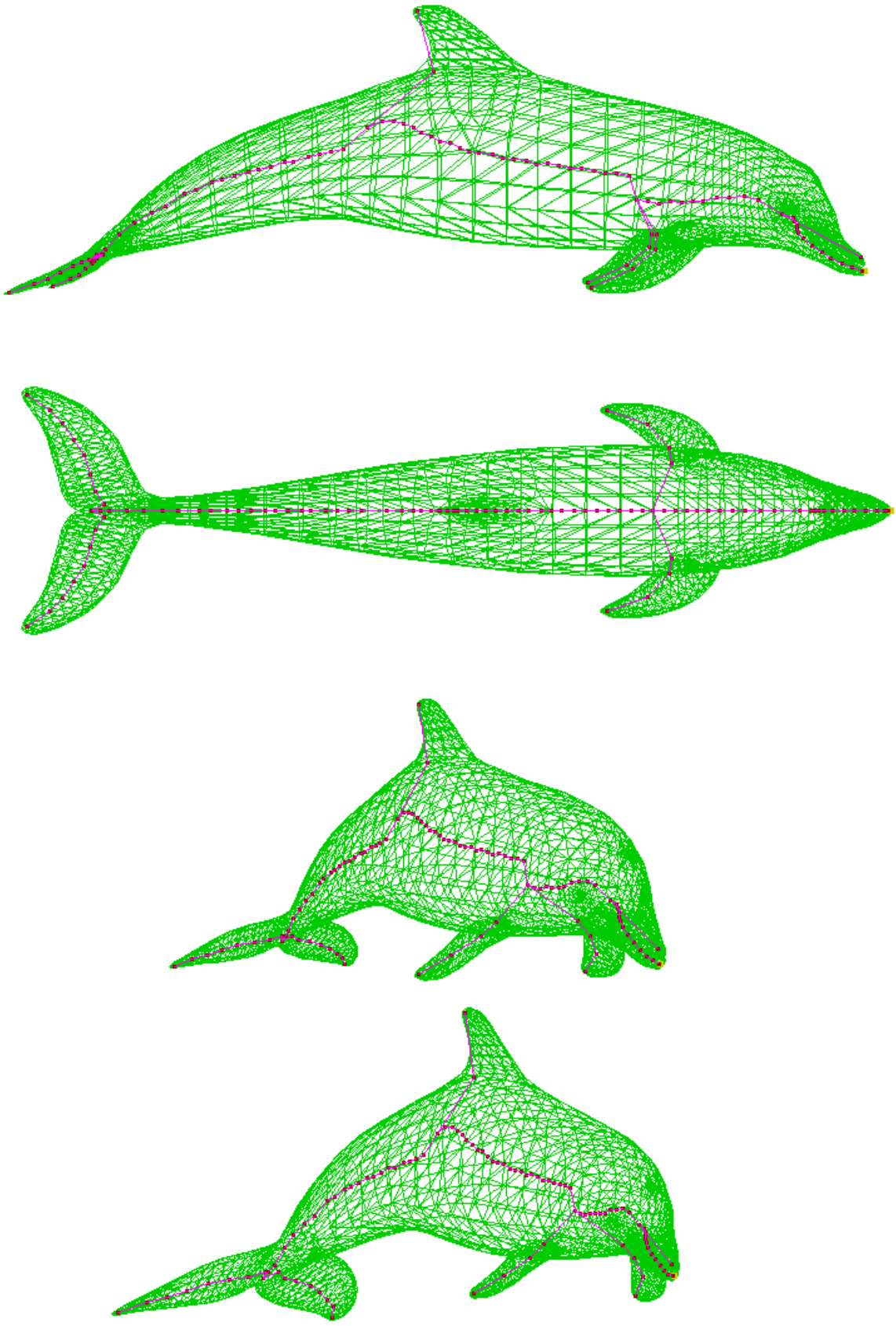


圖 16: 利用 Multi-Source 對 skeleton 做修正。

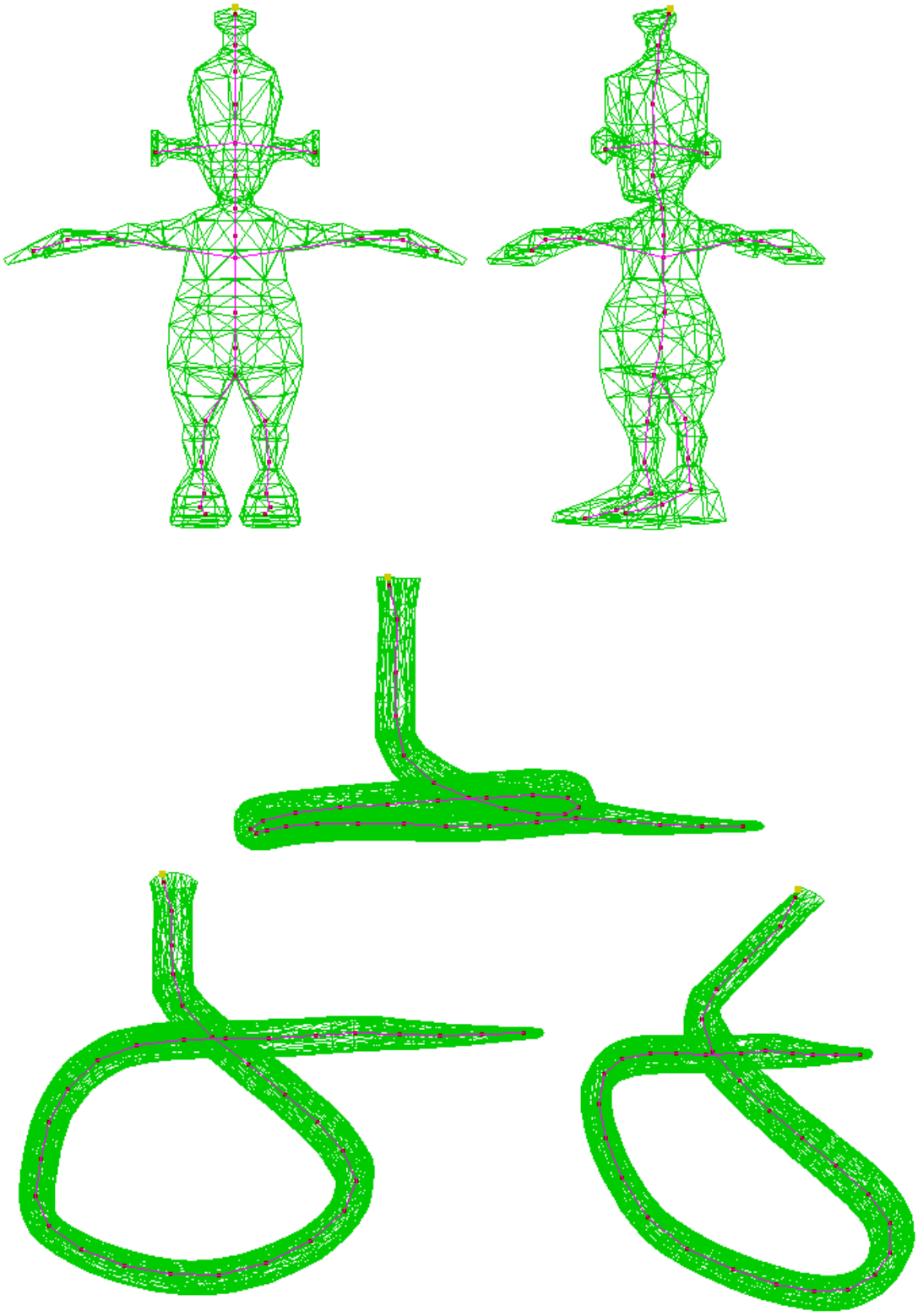


圖 17: 一些其他的模型。

2.2 Model Simplification

Skeleton 資料的計算，是我們的 levels of detail algorithm 的前置步驟。 Skeleton 跟 model 本身，可以說是一體兩面的資料。Surface model data 所代表的是 high resolution 下的模型資料，而 skeleton 則可以代表 lowermost resolution 的模型資料，兩者的關係相當密切。

一個好的 LOD 演算法，要考量很多方面的因素來決定物體的解析度，例如在場景內物體的遠近，大小，移動速度等。通常也跟人類在日常生活的視覺習慣有密切的關係，但必須具備幾點特質：首先，不能無限制的降低物體的解析度，也就是在做 LOD 時，對於構成物體的點，必須具備刪除”點”的下限。大多數在討論 levels of detail 的論文，都將重點放在刪去點的選擇與速度上的提昇 (triangulation)。而刪去點的選擇總是依循幾個方向，例如：利用視角與模型點的法向量來判斷 (viewer-dependence)，或是選擇最平坦的點 (此點與其周圍點的法向量差異度最小)，或是利用預先投影的方式，刪去投影後位置接近的點。這些觀點固然沒錯，但是大部分都只考慮局部區域，並沒有對整個模型做考量。然而，加入 skeleton 的資訊以後，我們將可以擁有更多的資料可以利用，讓我們更能掌握物件的整體形狀，更能確保物件形狀不被破壞。

尤其是在 lowermost resolution 底下，在此 level 下，因爲點數降到很低，所以重點就是如何保持模型的整體形狀，而對於較細部的地方就可以省略。雖然向量判斷法可以幫助我們選取出位於輪廓上的 vertices，但是還是需要很多 vertices 才能呈現出物件的外形。但是，如果我們有加入 skeleton 的資訊，我們將可以進一步

的降低 vertices 的數目，而且還能保持輪廓的完整，因爲我們可以利用 skeleton 的資訊，取出很平均分配的頂點，這也是我們演算法的最大優勢。

當我們要簡化模型的時候，我們考慮了下列幾個因素：

· I.Skeleton Section

首先，在建立 3D skeleton 的過程中，我們可以將所有的 vertices 的點集合 V 分爲兩個 subsets：其中一個集合 G，乃是屬於被使用來生成 skeleton 的 Geodesic Graph，由於 skeleton 分爲 k 個 sections 的關係，集合 G 又可分爲 k 個 subset：G(1)、...、G(k)。另一個集合 V-G，則也可分爲 k 個 subset：M(1)、…、M(k)，每一個 subset 都落在某一段 skeleton section 的中間。我們根據這些 information 來進行初步的 vertex clustering，並且對每一個 subset 計算並記錄下其所屬的 vertices 之 index。

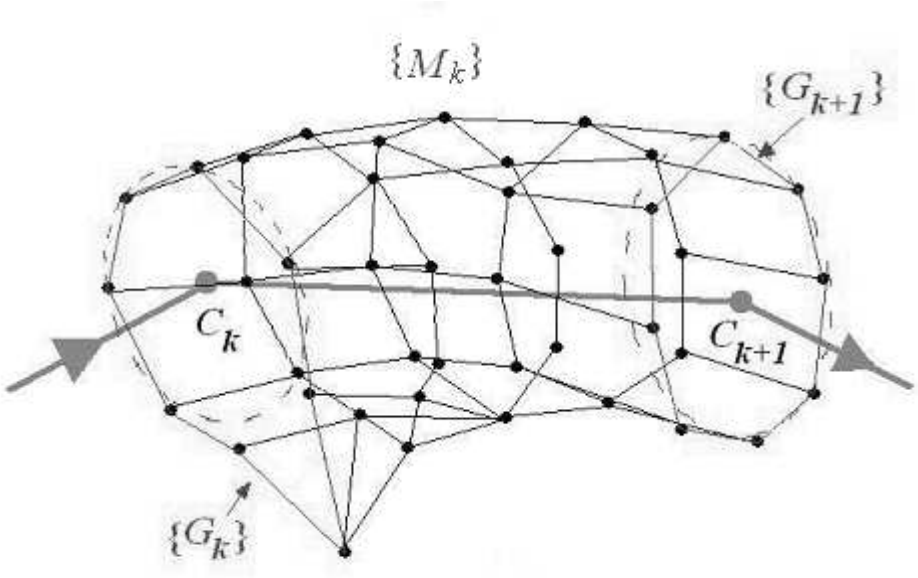


圖 18: G 與 M 集合的關係。

透過將模型的 vertices 分成兩個 Sets 的過程，我們可以知道，其中 G 是比較

重要的一群點。因爲它們可以代表整個模型的主要形狀，當然，skeleton 若是取的不好，則 G 所呈現的外型就會跟原本的形狀有差距。所以，由此可知， G 中的點是比較重要的，當我們在選取點的時候應該優先保留。而剩下的點集合 M 給予較低的權重，會優先被刪除。

· II.Screen-Space Projection

決定是否要 Merge 掉一個 edge 應該要依據它在螢幕上的投影長度而定，而不是依照它在 3 維座標中的邊長。這也是 LOD 最基本的要求，因爲如果兩個點在螢幕上的投影點非常近，甚至重疊，我們就只要取其中一點就可以表示，並不需要多浪費時間再去計算另一個重疊的點。所以，我們設定一個 threshold，當螢幕上的兩個投影點距離小於 threshold 時，我們就只取其中一點。我們的做法是，當某點所連結的 edges 在螢幕的投影長度小於 threshold 時，就減少它的重要性。所以，如果連結到某點的 edges 投影長度都很小，則這個點的重要性就會很低，在最後要選擇刪去點時，將會有很高的機會被優先刪去。

· III.View-Oriented

我們也加入了 View-Oriented 的要素，讓物件隨著觀測者位置的不同，來做調整。這個步驟必須要在 run-time 的時候執行。我們必須依照每個頂點的 normal vector(Vn) 與 " 頂點到 view 的向量 "(Vp)，兩個向量的內積來判斷。若 Vn 和 Vp 的內積小於 0，我們就可以知道此點位於物件的後方，就可以優先刪除； 若內積的值大於 0 且趨近於 0，則可以知道此頂點對於目前的觀測者而言，是位於物件的邊緣位置，則賦予它較高的權重，這樣就能夠盡量維持

住在邊緣附近的點。

要導入 View-Oriented 的要素，normal vectors 佔有重要的地位。若是 normal vectors 的值有誤差，會很容易出現誤差，做 LOD 時，容易在邊緣出現破片。每個頂點的 normal vector 應該要用連結到此點的三角形之 normal 的平均值，並參考三角形的面積給予不同的權重，面積越大者比重越高，這樣得到的平均法向量才是較正確的。

· **IV.Illumination**

考慮照明對於 LOD 的影響，是一個很好的構想。我們可以在模型上，亮度有明顯改變的地方用較高的 resolution 來表現，我們也可以藉此讓高亮度的地方有較好的解析度。因爲我們考慮到有亮度的部分比較會吸引觀測者的目光，所以提高解析度是合宜的，也讓我們的 LOD 系統更爲逼真。

在經過上述的條件判斷以後，我們並沒有立刻決定要刪除的點，而是給予每個頂點一個值，值越大表示此點的重要性越高，越小則表示重要性越低。經過一個距離值 (criterion) 的調整， criterion 是依照距離來調整，根據表面積與距離的關係我們可以知道，面積與距離的平方成反比，而面積也是代表我們需要用多少點數來表示模型，所以，

$$resolution \simeq \frac{1}{criterion^2}$$

我們根據得到的點數來選擇要刪除的點。由上述關係可以得知，當距離變成原本的兩倍時，點數大約減少成爲原本的 1/4；距離變成原本的三倍時，點數大約減少成爲原本的 1/9。

2.3 Dynamic Triangulation

在決定了要刪去的點以後，接下來就要對模型重新三角化 (triangulation)。重新三角化的步驟是每個 frame 都要做一次，所以爲了提昇速度，我們必須有一些前置的資訊。由於 skeleton 的計算可以將模型分成兩部分 (集合 G&M)，所以我們可以預先計算出每個點在被刪去時應該要退化到的相對應點。

首先，對於集合 M，由於 M 集合是屬於較容易被刪去點，所以，我們直接計算每個在 M(k) 中的點，把它的退化點對應到 G(k-1) 和 G(k)，從 G(k-1) 和 G(k) 中選出離 M(k) 中的點最近之相對應點並記錄下來。在 triangulation 的步驟時，若 Mk 中的點需要被刪去時，我們只要重新搜尋一次所有的 triangles，並將要刪去點的 index 換成已記錄的退化點之 index，再去除不正確的三角形，就可以完成 triangulation。

同理，對於集合 G，對於同屬於 G(k) 的所有點我們都必須先記錄它的退化順序 (degenerate list)，用來儲存每個點要被刪去時應該要被合併到的另一點，這個 degenerate list 記錄的 index 總是同屬於 G(k) 的其他點。用 list 的原因是爲了避免同時刪去太多點時所造成的錯誤，例如，要同時刪去 a 和 b 兩點，而 a 的退化點是 b，但是 b 的退化點也是 a，這樣會造成程式的錯誤。爲了避免這種狀況，我們必須對每個點建立一個 list(依據距離來建立 degenerate list)，並且確保每個 G 的子集合都至少保留一個點。

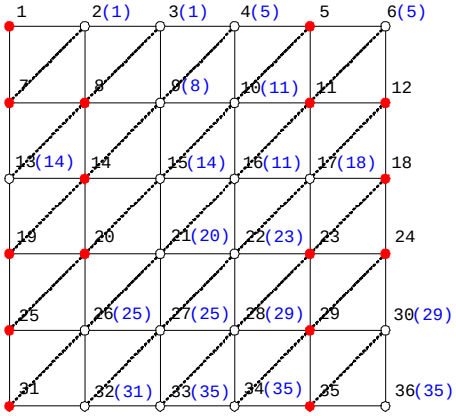
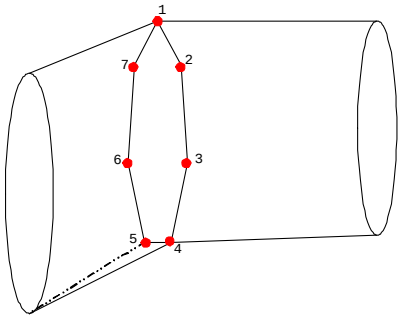
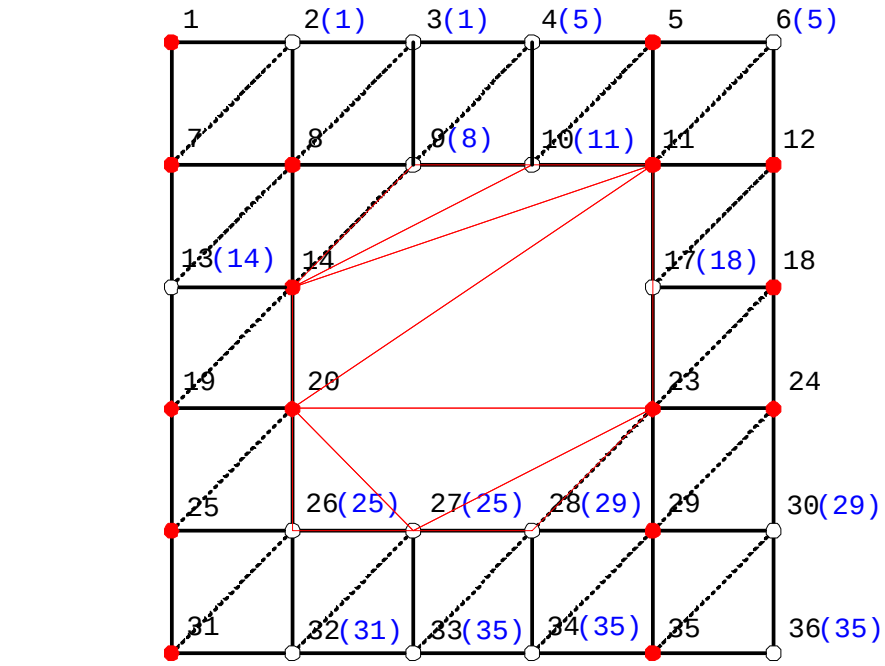


圖 19: 每個 M(k) 中的點都記錄一個退化點。




```
1 : 2 -> 7 -> 3 -> 6 -> 4 -> 5
2 : 1 -> 7 -> 3 -> 4 -> 6 -> 5
3 : 4 -> 5 -> 2 -> 1 -> 6 -> 7
4 : 5 . . . . . .
.
.
.
```

圖 20: degenerate list。



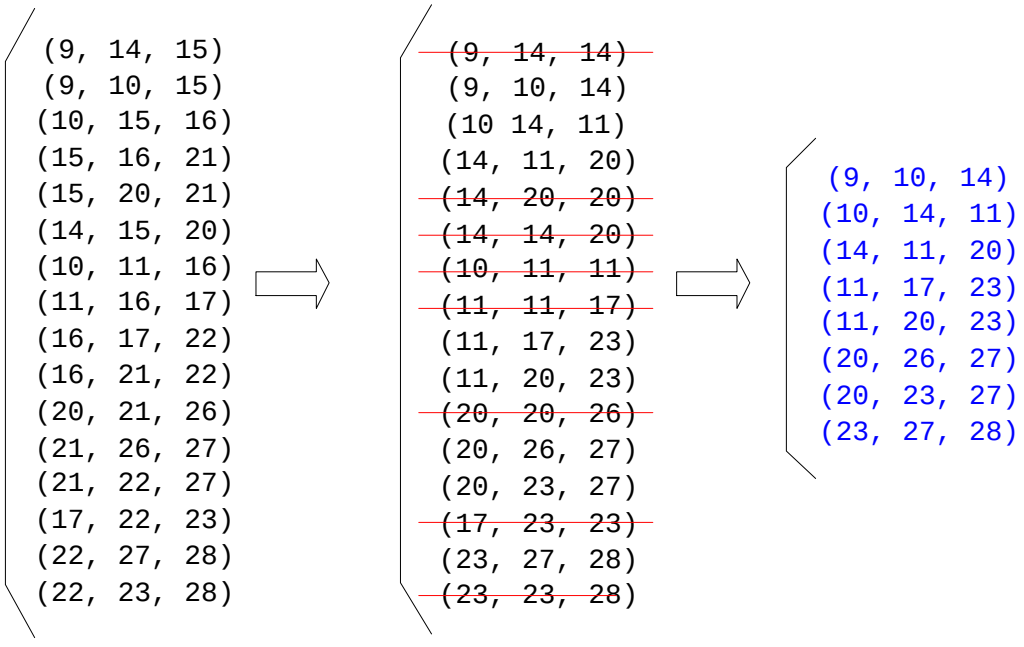


圖 21: 執行 reTriangulation 時可加快速度。

G	491	491	491	370	323	305	273	212
M	587	412	254	236	221	106	34	17

圖 22: 詳細的數據，對應下圖。

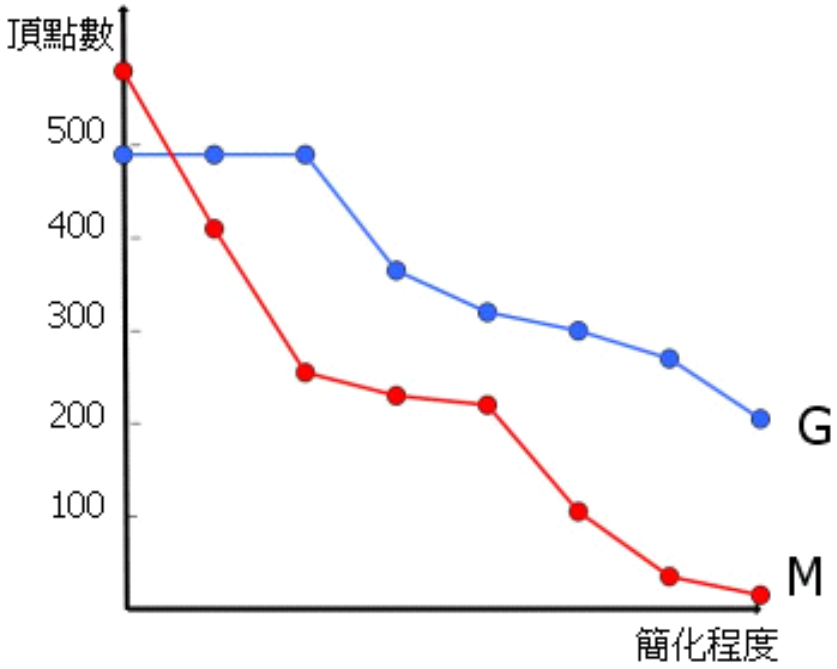


圖 23: 集合 G 和集合 M 的點數的比較 (使用海豚的模型時)，由圖可知，集合 M 的衰退速度遠遠大於集合 G。

3 Conclusion and Result

我們已經成功的將 3D skeleton 導入 LOD 系統之中，由以下的圖 24與圖 25的比較中，我們可以發現，當我們只採用 3D skeleton 所對應到的模型上的點來建構我們新的模型時，物件的外觀仍然保持的相當完整，但是已經可以降低將近一半的點數，當然減少的點數是可以由 3D skeleton 步驟的 k 值來決定。若 k 值取的越大，也就是說我們用來取樣 3D skeleton 的 level 越多，則得到的相對應點就越多，那麼在 LOD 的步驟時所以刪去的點就越少，以目前來說，我們採用的方式約可以刪去一半的點。此外，我們可以看出，大部分被刪去的點都是比較密集部分的點，這是因爲我們在取樣 3D skeleton 的 level 時所用的間隔是固定的，所以只有用 skeleton 所對應的 vertex 所建構成的 model 跟原來的模型比較起來，我們可以發現原來的模型被”平均化”了，在點分布密集的地方解析度被降低了，而在點的分布比較不密集的部分幾乎沒有變動，當然，這樣的結果是跟 LOD 的理念相符合的。但若是我們想要改變這樣的結果，只要在取樣 3D skeleton 的 level 時，隨著當時的模型複雜度來改變取樣的間隔就可以達成了，點密的地方用較短的間距，點稀疏的地方使用較長的間距。在圖 28和圖 30中我們放入當時點數所對應的距離所投影的圖形，讓讀者能夠比較。

我們的執行環境爲：AMD Althlon(tm)1.2G，256M RAM，GeForce2 MX100/200，程式使用 visual C++(using MFC) 撰寫。

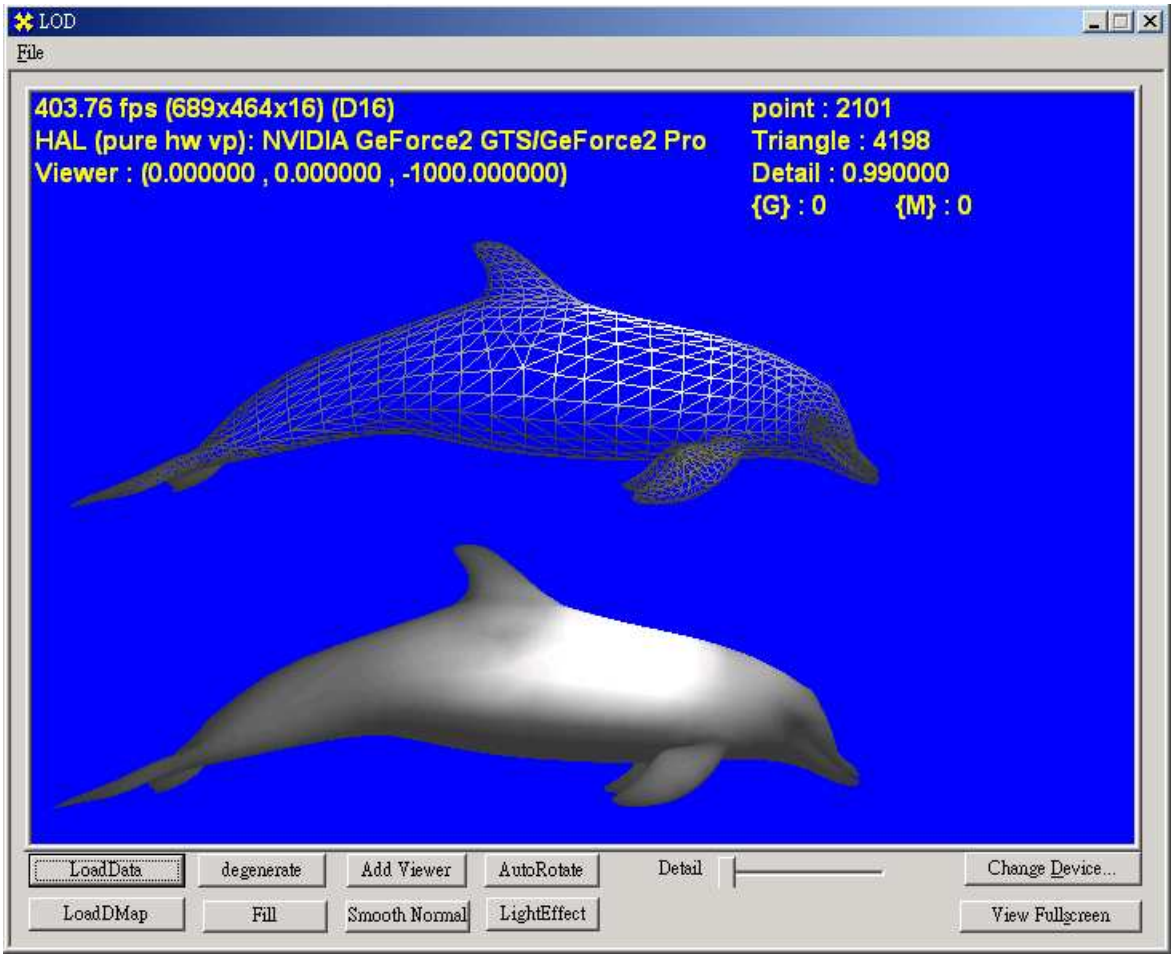


圖 24: 海豚 : 原本的模型，有 2101 個頂點與 4198 個三角形。

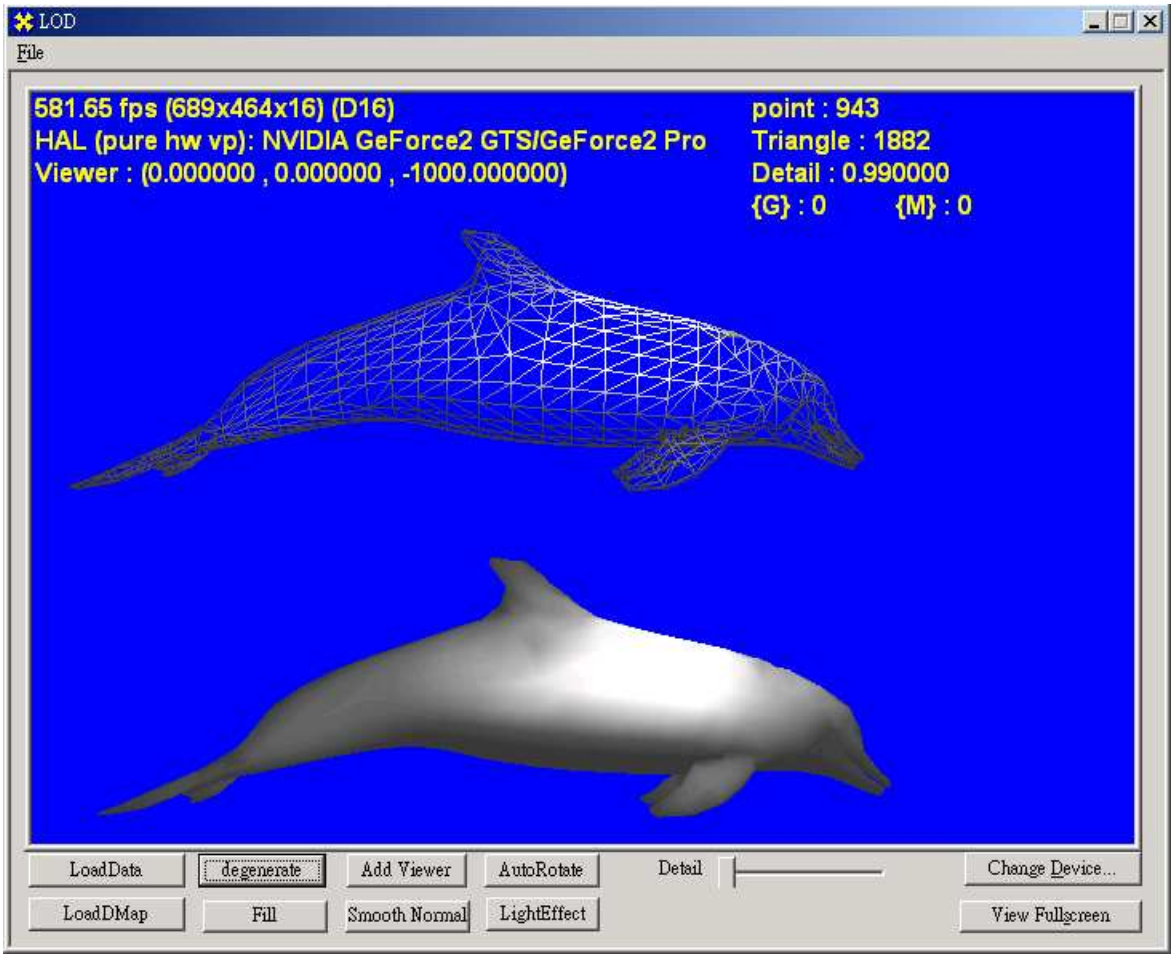


圖 25: 海豚 : 只採用集合 G 所包含的頂點時，有 943 個頂點與 1882 個三角形。

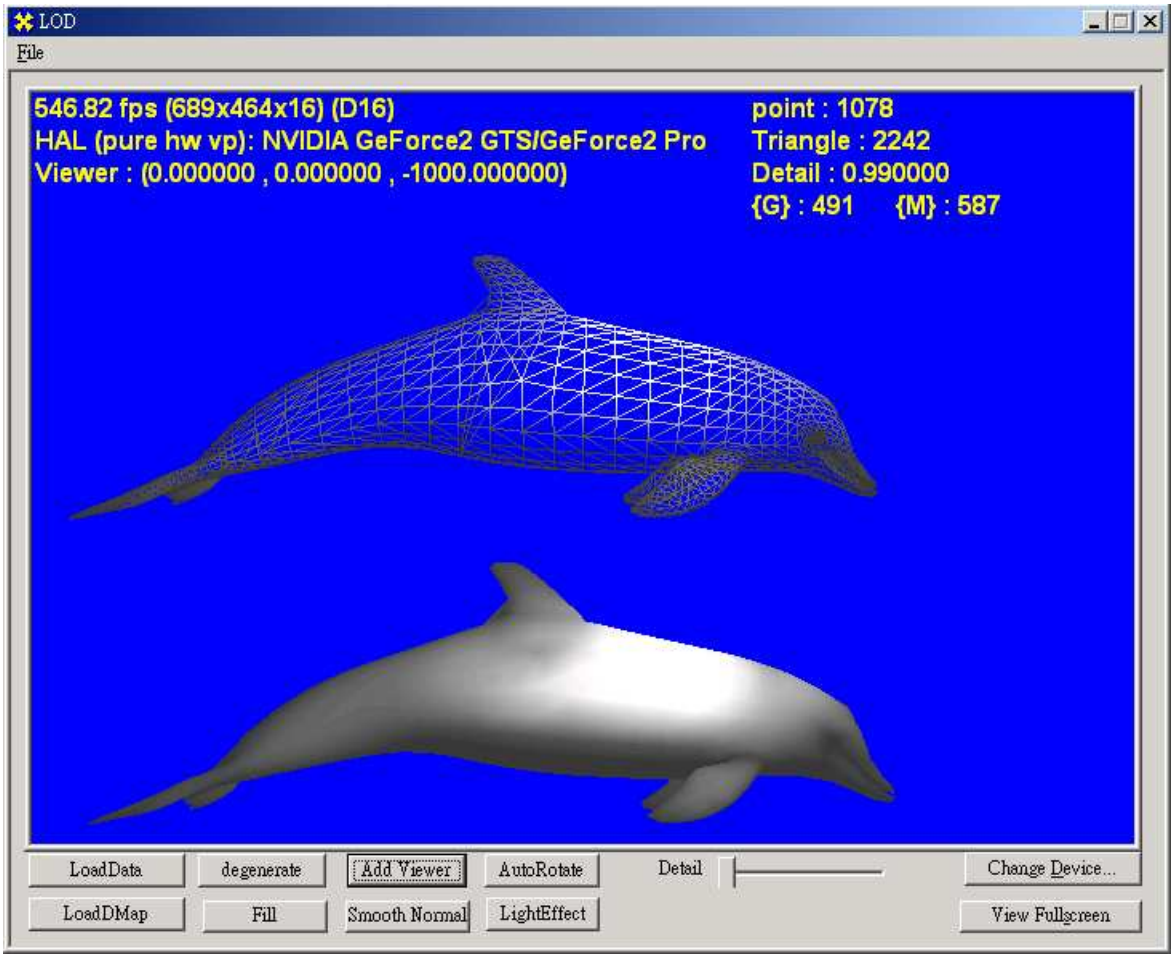


圖 26: 海豚 : 較高解析度下的 LOD，有 1078 個頂點與 2242 個三角形。

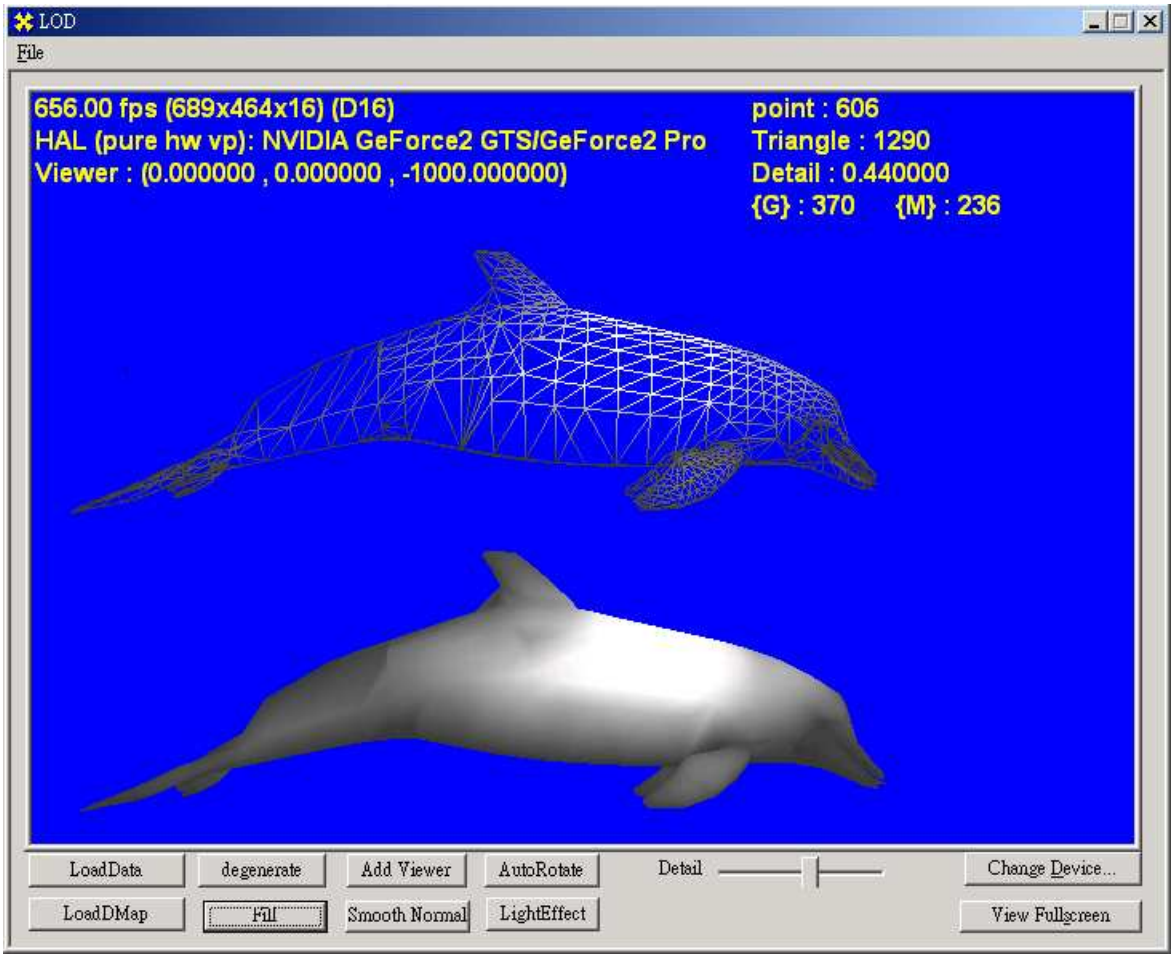


圖 27: 海豚 : 大約 2 倍距離下，有 606 個頂點與 1290 個三角形。

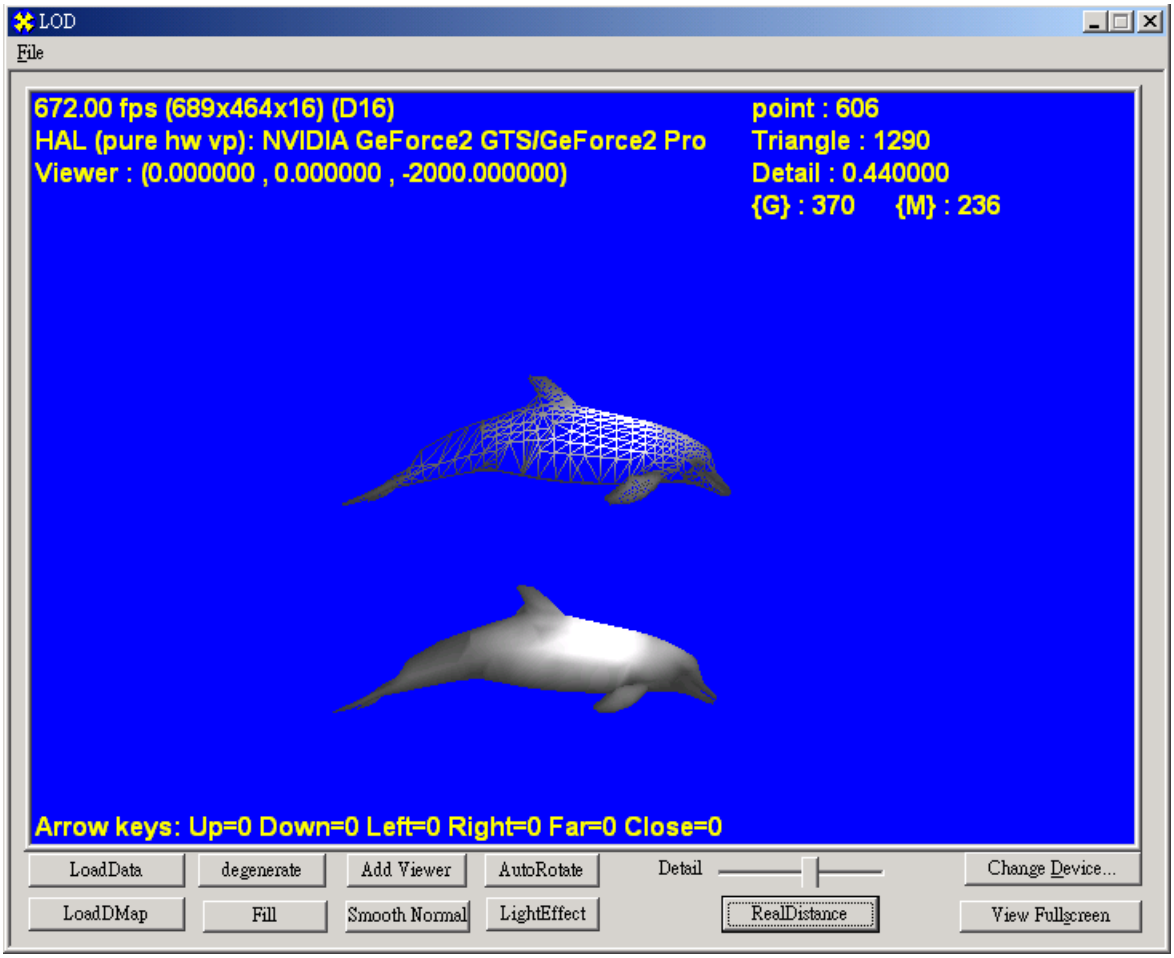


圖 28: 海豚 : 大約 2 倍距離下，實際的投影畫面。

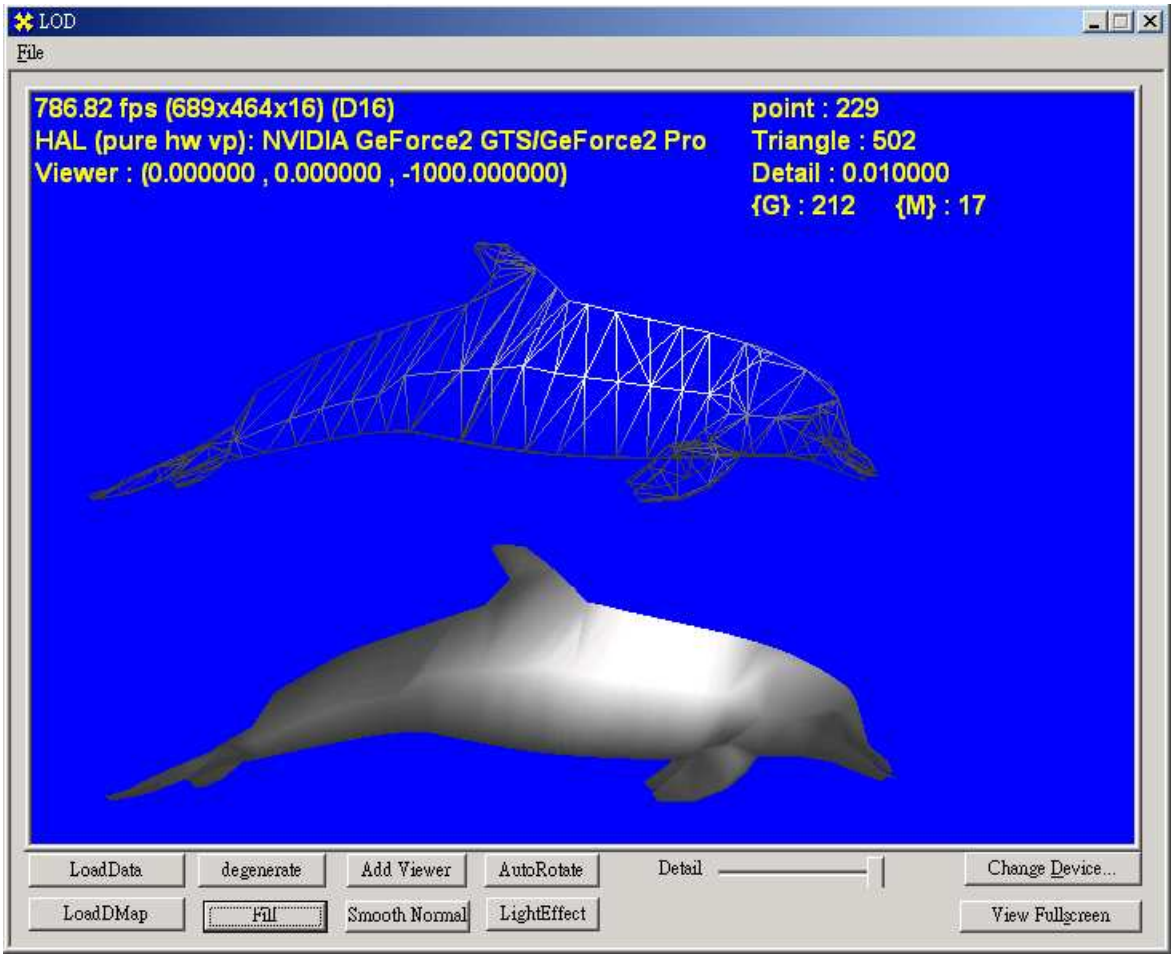


圖 29: 海豚 : 大約 3 倍距離下，有 229 個頂點與 502 個三角形。

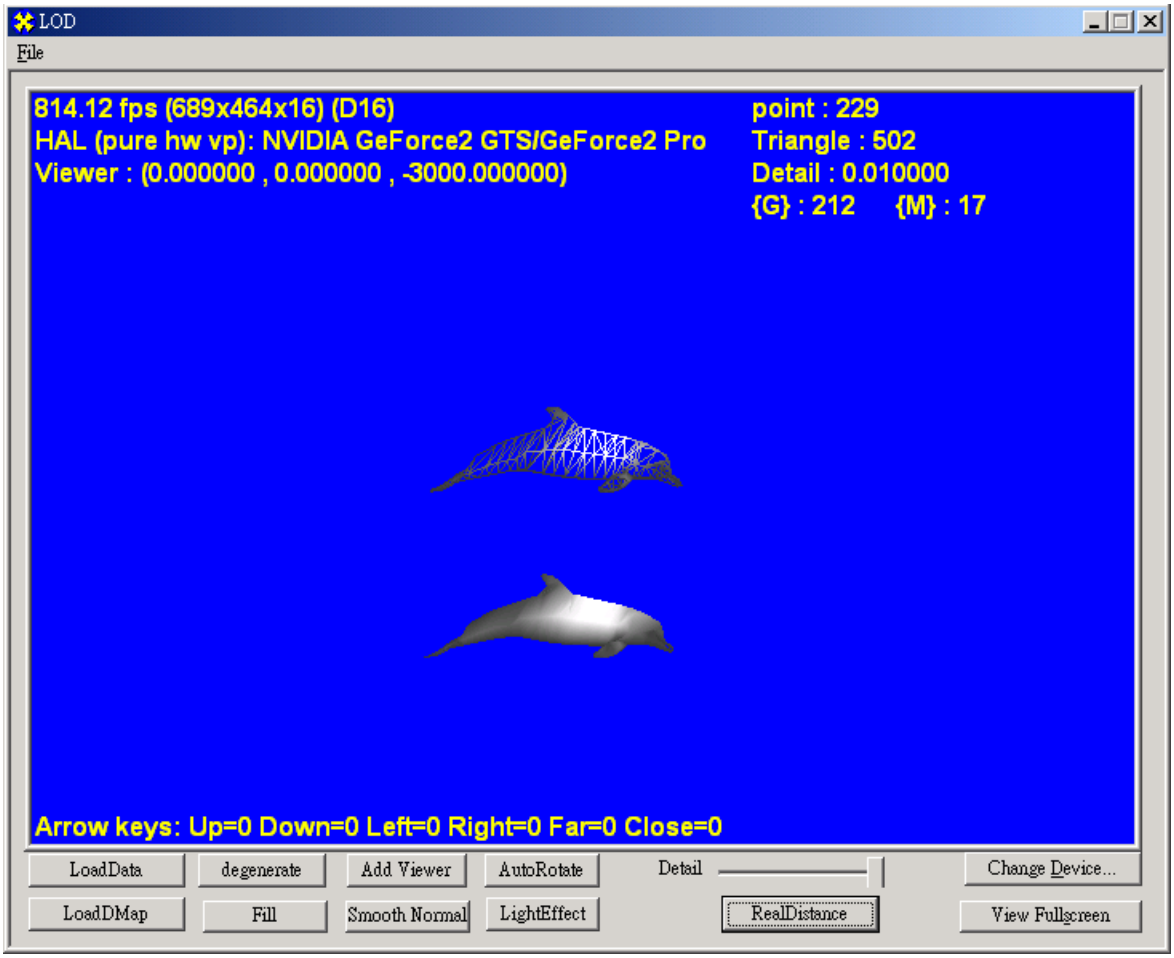


圖 30: 海豚 : 大約 3 倍距離下，實際的投影畫面。

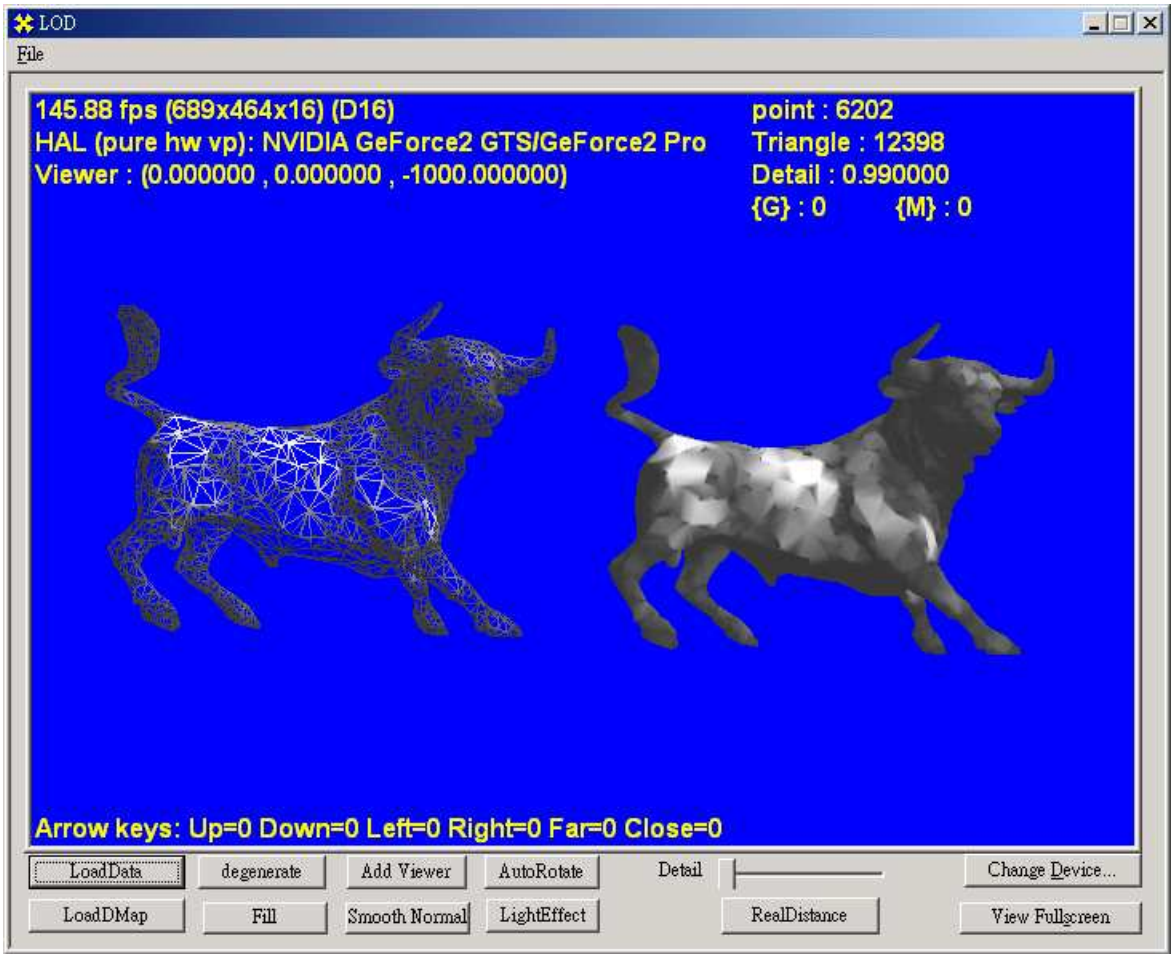


圖 31: 牛 : 原本的模型，有 6202 個頂點與 12398 個三角形。

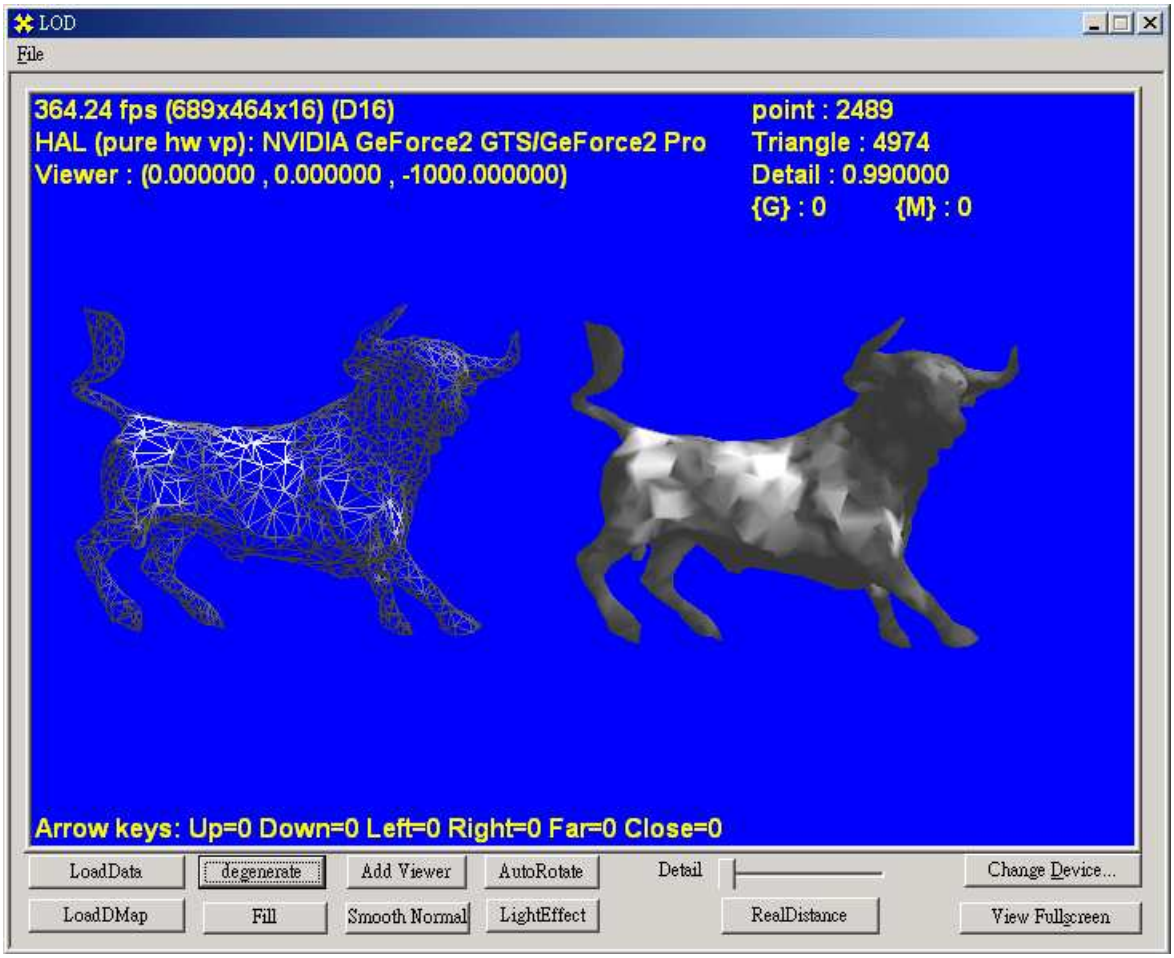


圖 32: 牛 : 只採用集合 G 所包含的頂點時，有 2489 個頂點與 4974 個三角形。

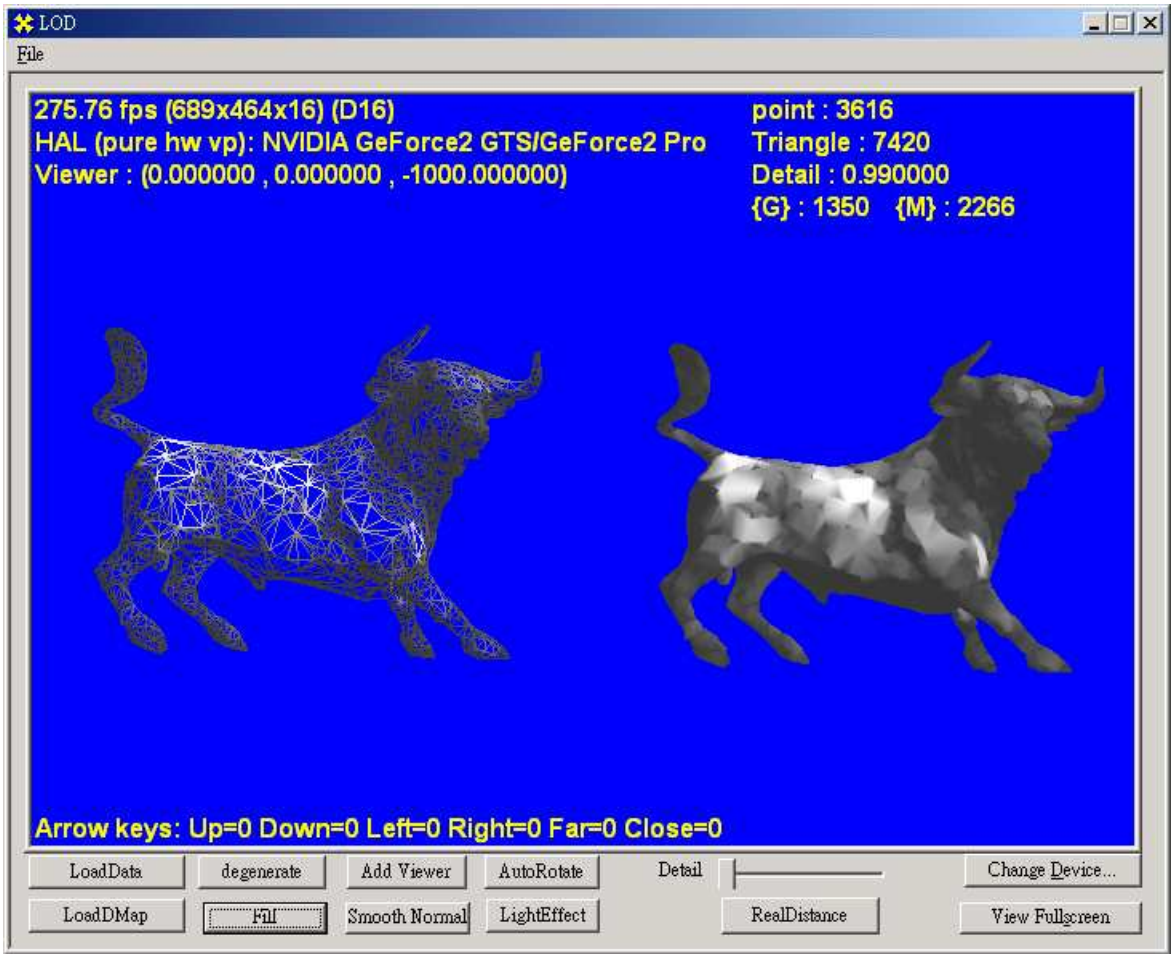


圖 33: 牛 : 較高解析度下的 LOD，有 3616 個頂點與 7420 個三角形。

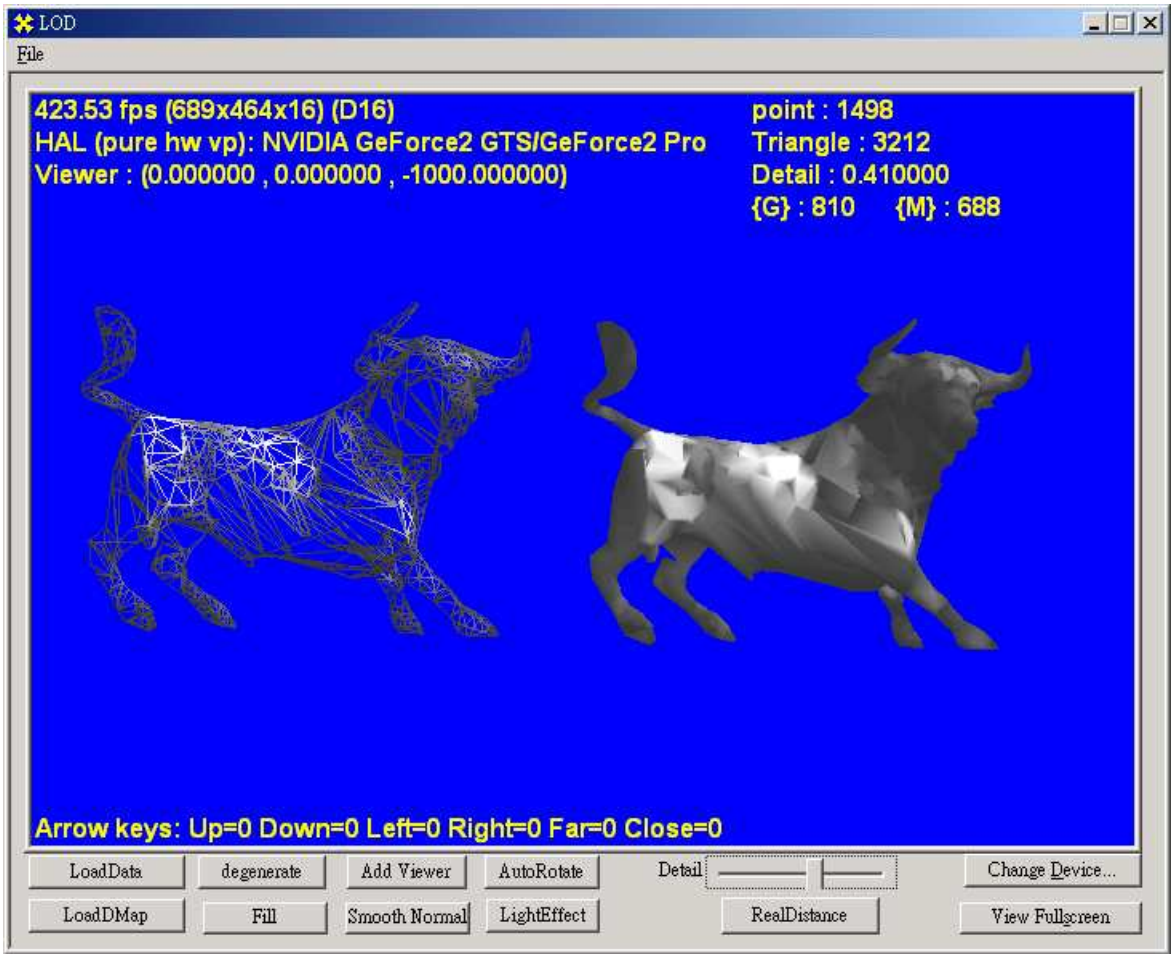


圖 34: 牛 : 大約 2 倍距離下，有 1498 個頂點與 3212 個三角形。

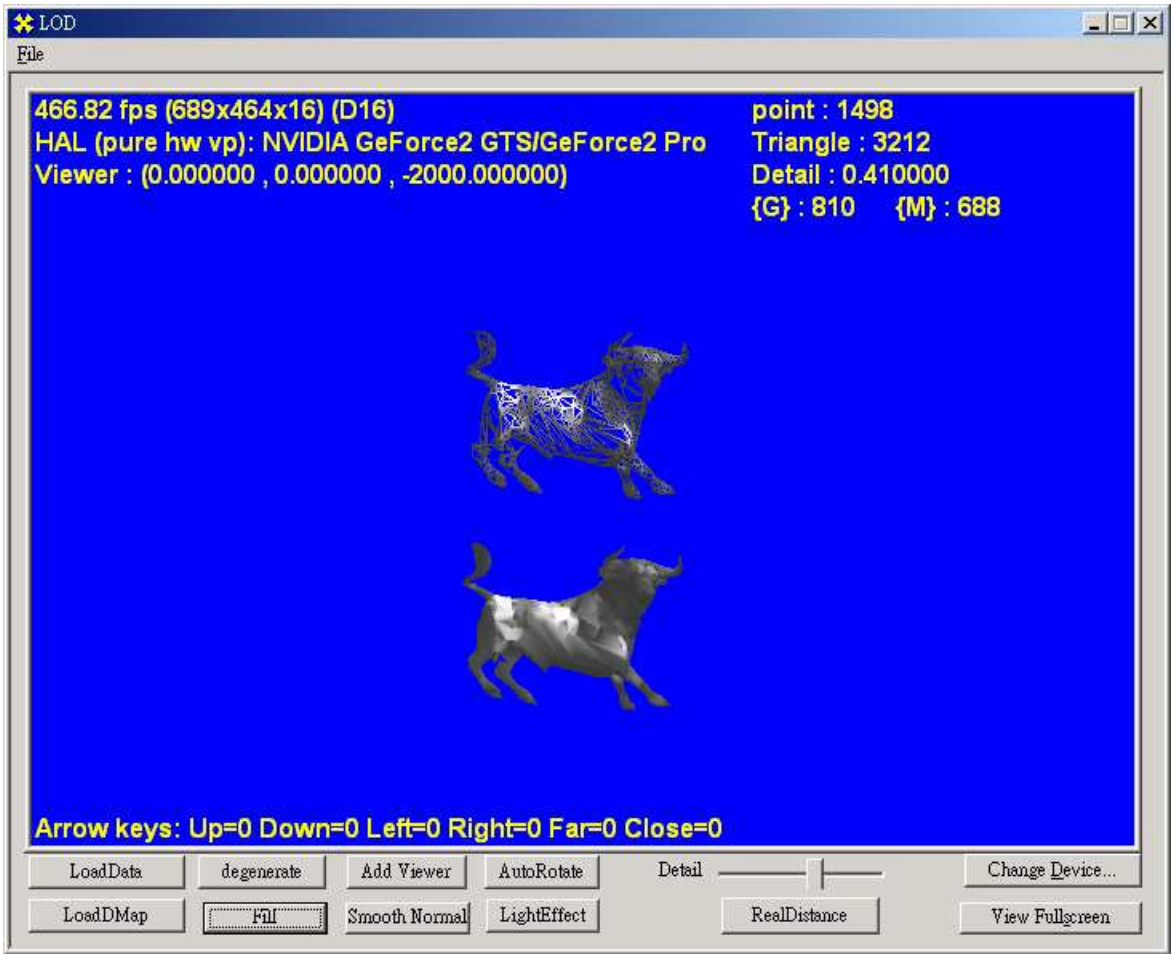


圖 35: 牛 : 大約 2 倍距離下，實際的投影畫面。

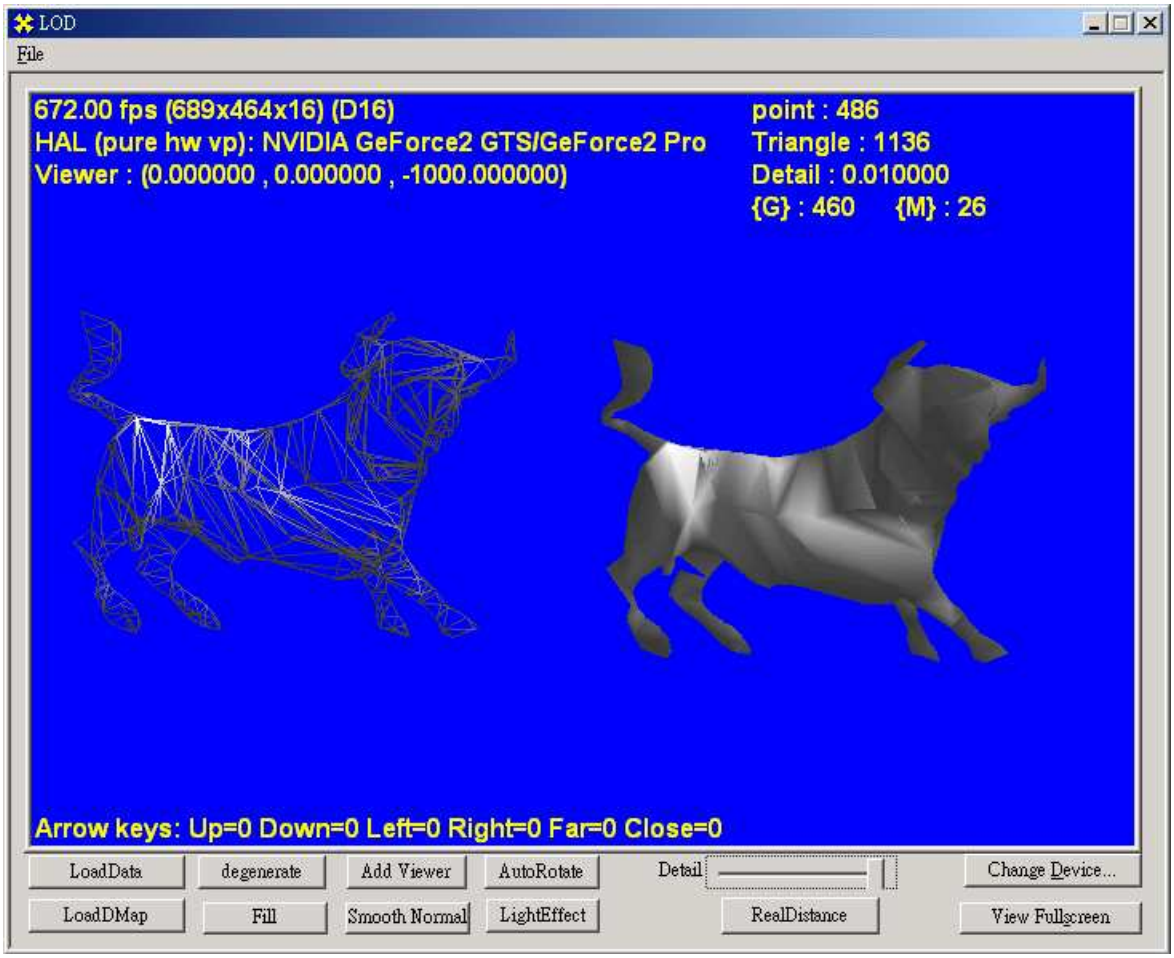


圖 36: 牛 : 大約 3 倍距離下，有 486 個頂點與 1136 個三角形。

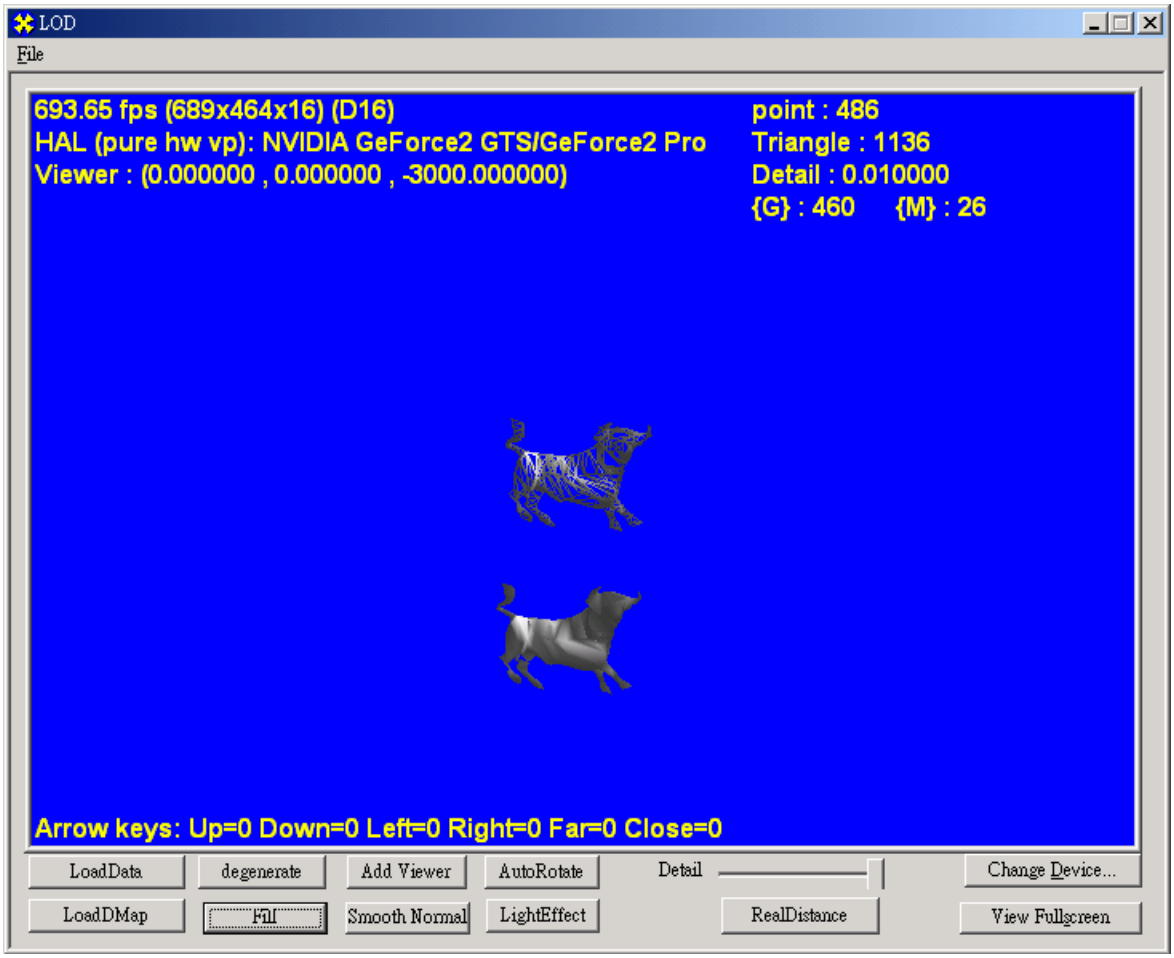


圖 37: 牛 : 大約 3 倍距離下，實際的投影畫面。

4 Future Work

到目前爲止，我們所提出的演算法只算是初步的結論，雖然我們已經成功驗證了 skeleton 對於 LOD 有莫大的幫助，並且實做出一個 LOD 系統，但是尙有很多議題需要討論與解決：

Model optimization

模型的複雜度會影響我們演算法的速度與效能，一個物件如果 branches 太多，會讓我們的演算法效能降低，所以我們需要一個物件的前置運算系統，這個物件前置運算系統能將複雜的模型切割成若干個較小且較簡單的形狀，我們再分別對這若干個切割後的模型做 skeleton。這樣，不但 skeleton 的結果較爲正確，而且對於提昇本演算法的效能與品質亦有莫大的幫助。就理論上而言，我們的演算法對於狹長型的物件有最好的效率，所以，透過物件的事先計算分割，來加快與提高演算法的應用性，是值得且有其必要的。而且，這個物件前置運算系統是可以完全獨立於我們的演算法之外。

3D skeleton algorithm

我們目前所使用的 3D skeleton algorithm 仍然有其缺點，其一，source point 的選擇對於 skeleton 會有相當程度的影響，特別是在 branches 的地方；其二，用 k 值取 level 的時候因爲使用固定的距離，所以對於有 branches 的地方無法加強其重要性。目前想到的方法是用 multi-pass 的方式，先使用正常的 3D skeleton algorithm 對物件運算一次，找出 branches 的部分並記錄之，接著再重新運算一次，並且在 branches 的部分再做細分，以便可以得到充分的資訊，

此外，也利用 multi-source 的方式，來校正 skeleton。

Vertex merge problem

目前我們所使用的 triangulation 方式仍嫌不足，有少許的機率會出現破片的問題，這部分還有改善的空間，此外，目前的系統中尚未加入避免 jump 的技術，所以畫面時有跳動的情況，這也是未來努力的方向之一。

參數式曲面資料結構的導入

目前我們所使用的物件都是 polygon model，我們希望在未來支援一些參數式的模型，讓我們的系統更加完善。

參考書目

[1] Anne Verroust, and Francis LaZarus. Extracting Skeletal Curves from 3D Scattered Data. September 1997.

[2] D. Attali and A. Montanvert. Computing and simplifying 2D and 3D continuous skeletons. Computing Vision and Image Understanding, 67(3):261- 273, 1997.

[3] H. Sundar, D. Silver, N. Gagvani and S. Dickinson. Skeleton Based Shape Matching and Retrieval. Shape Modelling and Applications Conference, SMI 2003, Seoul, Korea, May 2003.

[4] M. Mortara and G. Patan'e. Affine-invariant skeleton of 3D shapes. Proceedings of the Shape Modeling International 2002(SMI'02).

[5] Silvia Biasotti, Simone Marini, Michela Mortara, Giuseppe Patan'e. An Overview on Properties and Efficacy of Topological Skeletons in Shape Modelling. Proceedings of the Shape Modeling International 2003(SMI'03).

[6] J.-H. Chuang, C.-H. Tsai, and M.-C. Ko. Skeletonization of Three-Dimensional Object Using Generalized Potential Field. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1241-1251, 2000.

[7] W.-C. Ma, F.-C. Wu, and M. Ouhyoung. Skeleton Extraction of 3D Objects with Radial Basis Functions. Proceedings of Shape Modeling International 2003(to appear).

[8] F.-C. Wu, W.-C. Ma, P.-C. Liou, R.-H. Liang and M. Ouhyoung. Skeleton Extraction of 3D objects with Visible Repulsive Force. Eurographics Symposium on Geometry Processing(2003).

[9] Daniel Cohen-Or and Yishay Levononi. Temporal continuity of levels of detail in delaunay triangulated terrain. In Proc. Visualization 96, pages 37-42. IEEE Computing Society Press, 1996.

[10] M. de Berg and K. Dobrindt. On levels of detail in terrains. In Proceedings 11th ACM Symposium on Computational Geometry, pages C26–C27, Vancouver (Canada), 1995. ACM Press.

[11] Peter Lindstrom, David Koller, William Ribarsky, Larry F. Hughes, Nick Faust and Gregory Turner. Real-Time, Continuous Level of Detail Rendering of Height Fields. Proceedings of SIGGRAPH 96, pp 109-118, 1996.

[12] Stefan Roettger, Wolfgang Heidrich, Philipp Slusallek, and Hans-Peter Seidel. Real-Time Generation of Continuous Levels of Detail for Height Fields. In V. Skala, editor, Proc. WSCG '98, pages 315-322, 1998.

[13] Chew, L. P. "Constrained Delaunay Triangulations," Proc. 1987. Third Annual ACM Symposium on Computational Geometry.

[14] Sumanta Guha. An optimal mesh computer algorithm for constrained Delaunay triangulation. In Proceedings of the 8th International Parallel Processing Symposium, pages 102–109. IEEE, April 1994.

[15] Hoppe, H. Progressive Meshes. Computer Graphics(SIGGRAPH,96 Proceedings) (1996), 99-108.

[16] H. Hoppe. Efficient implementation of progressive meshes. Computers & Graphics, Vol. 22, No. 1, 1998, pages 27-36.

[17] H. Hoppe. View dependent refinement of progressive meshes. In ACM SIGGRAPH Conference Proceedings, pages 189–198, 1997.

[18] H. Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. IEEE Visualization 1998, October 1998, pages 35-42.

[19] Xia, J.C.; El-Sana, J.; Varshney, A. Adaptive real-time level-of-detail based rendering for polygonal models. Visualization and Computer Graphics, IEEE Transactions on , Volume: 3 Issue: 2 , April-June 1997 Page(s): 171 -183