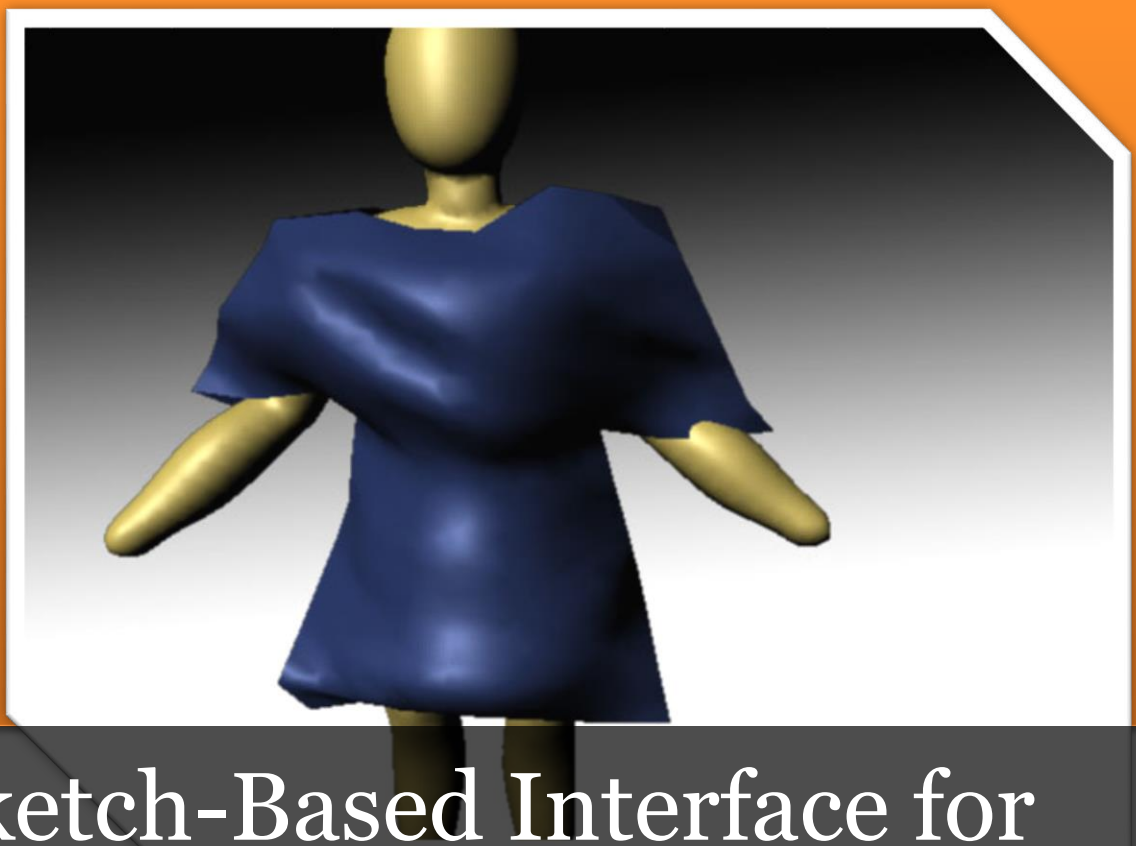


A Sketch-Based Interface for Clothing Design

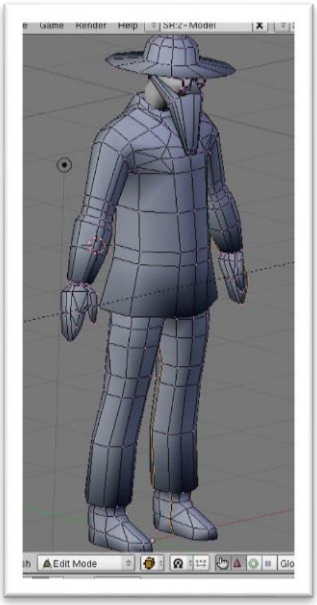
B95201019 數學四 劉光鑫

B96705003 資管四 王珣沛

B97706037 資管三 陳少祁

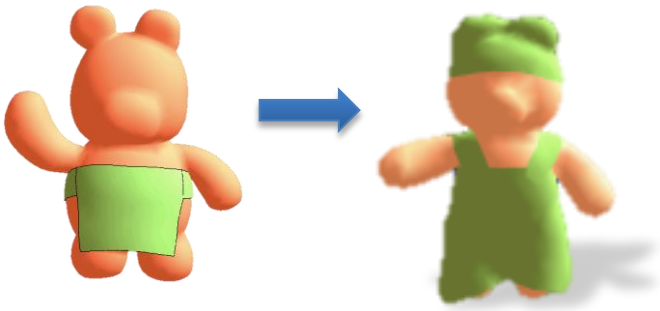
一、研究動機

我們想要提供一套可以讓服裝設計者輕鬆設計衣服的系統（fashion design），快速的讓服裝設計的創意轉變為 3D 雛型，讓設計者可以更輕鬆地檢視並修改其創意。但我們發現，使用傳統的方法要在 3D Model 上面加衣服一直是很複雜的事情，過去多是依賴美工在做角色 Modeling 的時候直接在角色身上做出凸起與凹陷的效果以模擬出衣服的樣子（如圖一）。這種方法使得 Model 和衣服成為不可分割的一體，設計角色時就必須先考慮其服裝，一旦角色設計完成便不太可能為其更換不同的衣服，且角色與服裝的設計過程耗時且複雜，因此不適合服裝設計者使用。



圖一　利用傳統建模軟體建造角色服裝

我們希望可以讓服裝設計者以最輕鬆無負擔的方式製作出服裝的 3D 雛型，而最直覺的方式就屬手繪了，因此我們想要設計出一套使用者只要畫出 2D 平面的衣服輪廓線就可以自動生成 3D 衣服雛型的系統，協助服裝設計者盡情地發揮創意。



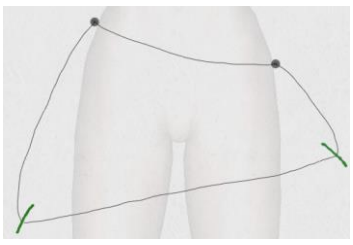
二、參考論文

- Emmanuel Turquin， Marie-paule Cani， John F. Hughes.: **Sketching garments for virtual characters**, Published in Proceeding SIGGRAPH '07 ACM SIGGRAPH 2007 courses
- IGARASHI T., HUGHES J. F.: **Clothing manipulation**. In Proceedings of the 15th annual ACM symposium on User interface software and technology (2002), ACM Press, pp. 91–100.

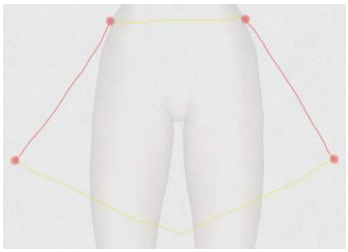
三、Algorithm

1) 對使用者輸入的 2D 設計圖形做處理

— 使用者在螢幕上輸入服裝設計圖的輪廓（見下圖）。



— 將使用者輸入的輪廓線條區分為 border lines（線條有經過 body 者，下圖中黃線）與 silhouette lines（線條沒有劃過 body，下圖中紅線）兩大類。

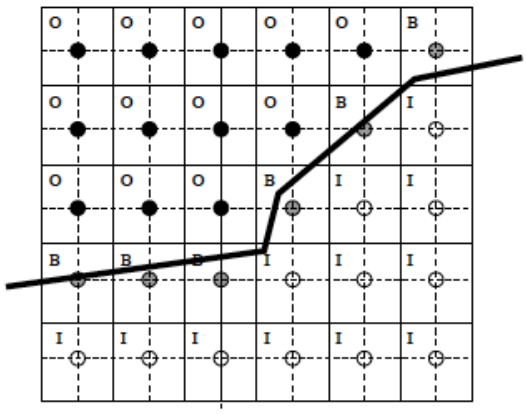


2) 將 2D 設計圖形做初步的 3D 運算，找出所需之資訊，並計算出輪廓線上到 body 的距離值 d:

- 對使用者輸入的各個點找出其點到 body 的距離 d。輸入點落在 body 上者，其 d 值為 0；
- 落在 body 外的點，我們計算此點到投影過後的 2D body 輪廓圖形下的最短距離 d'，並將 body 上形成此最短距離之點 p 之 z 值指定給此輸入點。
- 對於每一條 silhouette line（線條不經過 body 者），我們內插出 silhouette 線上各點之 z 值， 並利用此 z 值，求出現上各點在 3D 座標系中到 body 的真正距離 d。
- 對於每一條 border line（線條有經過 body 者），我們使用線性內插內插出線上各點之 d 值。

3) 建立 3D 模型

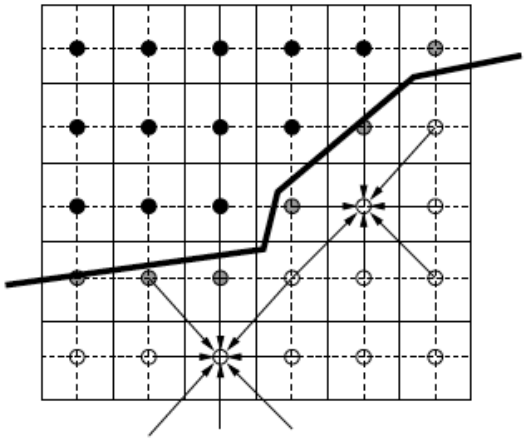
- 針對使用者輸入之設計圖形之區域做切割，切割出 16*16 的網格點。並根據設計圖形的輪廓線條，將網格點區分為：inside pixel (I)、 boundary pixel (B)跟 outside pixel (O)三種（如下圖所示）。



— 將輪廓線上的 d 值指定給鄰近對應的 boundary pixel，接著內插出 inside pixel 之 d 值。

— 對於每個 boundary 跟 inside pixel，找出 z 使得此點(x, y, z)到 body 的距離為 d。

— 對所有 boundary 及 inside pixel 之 z 值做 smoothing，使得整體曲線更加平滑（如下圖）。



— 最後連接鄰近之 boundary、inside pixel 成三角形，建立出 model。

四、實作細節

使用工具：

— **Qt 4.7 (使用者介面)**

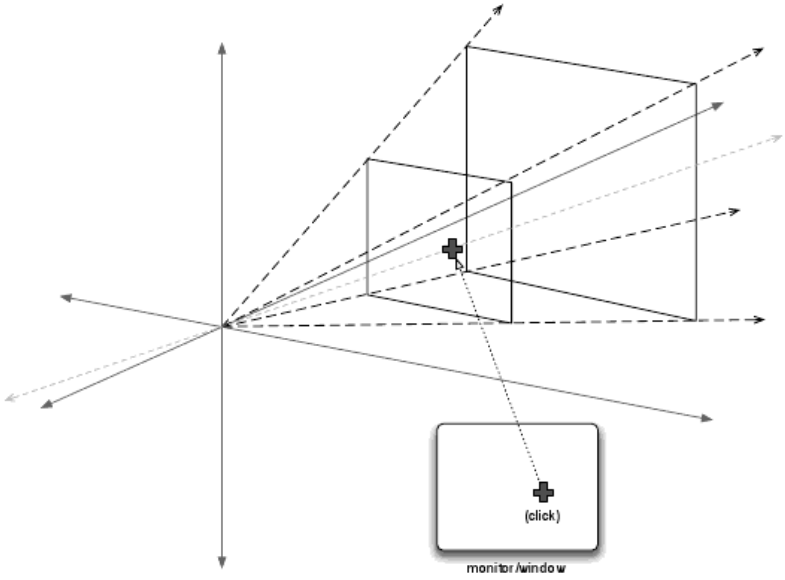
— **OpenGL**

— **OpenCV 2.2 (Delaunay Triangulation、判斷點是否在區域內)**

1) 對使用者輸入的 2D 設計圖形做處理

我們讀進使用者在視窗上點取之點座標，並利用 opengl 的 glUnProject()函式來找視窗平面上所選取的點，投影回 3D 座標空間的座標值。由投影回空間的座標(x, y, z)之

z 值，我們可以判斷此點是否有無落在 body 身上（若無，則回傳之 z 值會為 near/far z-clipping plane 之值）。



針對各個使用者輸入之輪廓點所組成之線段，若鄰近（上下一個）的輪廓點落在 body 上，則此一現段是為 border line；除此之外，針對前後輸入之鄰近輪廓點不落在 body 上者，我們將此一線段切成數個小段，依舊利用 UnProject 的方式，將這些小段端點投影回 3D 座標中，觀察其是否有落在 body 上者，若有，則將此前後鄰近輪廓點所組成之線段記為 border line；其餘線段則為沒有通過 body 之 silhouette line。

2) 將 2D 設計圖形做初步的 3D 運算，找出所需之資訊

針對各個使用者輸入之輪廓點，我們利用 1)所提及之 UnProject 方法所得之 z 值，來判斷此點是否落在 body 上。落在 body 上的輪廓點，我們將其點到 body 的距離 d 記為 0。而對那些沒有落在 body 上之輪廓點，我們利用以下之方法，找出此點到投影到 2D 平面上的 body 輪廓的最短距離 d'：

i) 我們先針對此輪廓點計算點到 body 上各三角形頂點的點到點之距離。

ii) 我們將每個三角形的各個邊投影到 2D 平面上，並利用向量、夾角、向量的投影長度來計算此輪廓點到線段的最短距離，並幫助判斷此點對此線段的垂直投影點是否會落在此一線段上（若無，則不考慮對此一線段的點到線段之距離）。

iii) 找出點到點以及點到線段距離之最短者 d'，並找出形成此最短距離之 body 上的相對應點（頂點或點到線段之垂直投影點）。注意，由於 2D 投影可能將空間中的兩個點投影到 2D 平面上的同一個點上，因此在計算最短距離並找出相對應點時，我們要記錄下（最短距離±ε）範圍內的所有對應點，並求出其平均。

找出輪廓點到 body2D 投影輪廓的最短之距離 d'之後，將造成此一最短距離 d'在 body 上之相對應點的 z 值指定給此輸入輪廓點。

對於每一條 silhouette，我們已經求出其組成之輪廓點之 z 值。針對此 silhouette 上之個點，我們對 z 值做線性內插，內插求出現上各點的 z 值。並針對各點(x, y, z)求出各點在 3D 空間中到 body 的真正距離 d。而點到 3Dmodel 之距離求法如下：

(a) 我們對 body 的各個頂點計算空間中點到點的距離。

(b) 針對每個 body 上的三角形，我們找出此三角形的單位 normal 空間向量，並依序對三角形的各個頂點找出頂點到我們目標點的空間座標向量 v。利用 normal 向量與向量 v 求內積、找出夾角，以及此點到三角形的垂直距離，並利用 normal 與 v 的夾角 α，以及 v 垂直投影倒三角形平面的投影向量 v'與此三角形頂點所夾之兩個邊的夾角 θ1、θ2，以及 v'之長度來判斷目標點對此三角形平面的垂直投影點，是否有落在此一三角形區域中。（若無，則忽略不考慮此一三角形所計算出來的點到平面的距離）

(c) 找出上述空間中點到點、點到三角形區域距離之最小者。並將此最小者記為空間中此點到 body 的距離 d。

而對每條 border line，我們現在有了線段兩端輪廓點的 d 值，因此，我們

僅需對此線段上的各點，針對 d 值做線性內插，求出各點相對應之 d 值。

現在，我們有了使用者輸入輪廓線上各點到 body 的距離 d。

iv) 建立 3D 模型：

我們根據使用者所繪製的設計輪廓找出一個適當的矩形區域（找出 minX, minY, MaxX, MaxY，再做稍微的放大，讓輪廓線不會剛好交到矩形區域的邊線），接著再等距切成 16x16 的網格。由於之後的計算是以各個網格的中點作為代表運算，因此我們只要記錄下個網格的中點(x, y)座標即可。

接著我們由使用者輸入的封閉輪廓線來將網格分類成 inside、outside。而輪廓線有與網格內水平、垂直座標軸（軸的交差點為網格的中心點）相交者，則記為 boundary。這個判斷一個點（我們對網格的運算判斷都以其網格的中心點來代表）在一指定輪廓內外的工作，可以使用 OpenCV 的 cvPointPolygonTest()函式來幫忙進行判斷。而 boundary 的判斷，則以中心點上、下、左、右分別平移半個單位距離，觀察其網格性質的變化（是否有 outside→inside，或 inside→outside）來判斷是否為 boundary。

再來，我們對每個 boundary pixel，我們找出此 boundary pixel 到使用者輸入輪廓線上的最近點，並由 iii)所算出來輪廓線上各點的 d 值，將此最近點的 d 值指定給此 boundary pixel。

填滿完每個 boundary pixel 的 d 值之後，針對 inside pixel 的部份，我們依據 inside pixel 到各個 boundary pixel 的距離平方之倒數做加權平均，求出此 inside pixel 所對應的 d 值。

然後，針對每個 boundary、inside pixel，我們在空間中，沿著 z 軸（垂直於螢幕平面）的方向，測試求出適當的 z 值，使得此 pixel 點(x, y, z)在空間中到 body 的距離為 d：

(a) 我們由 near z-clipping plane 出發，到 far z-clipping plane。我們先將一整段距離（near~far z-clipping plane）切成一百個分段。計算各個節點到 body 的距離 D，並將 | D - d | 小於此時分段單位間隔距離者 Seg 記錄下來。

(b) 若有 | D - d | < ε 者，則記錄下此時的 z 值。

(c) 當整段距離都跑過一回之後，開始將(a) 中所找到的所有 Seg(s)同樣切成 100 個分段，依照(a)、(b)的方法操作一遍。總共反覆運算 10 次。

(d) 同時在各步驟的運算當中，將 | D - d | 最小者所對應之 z 記錄下來。

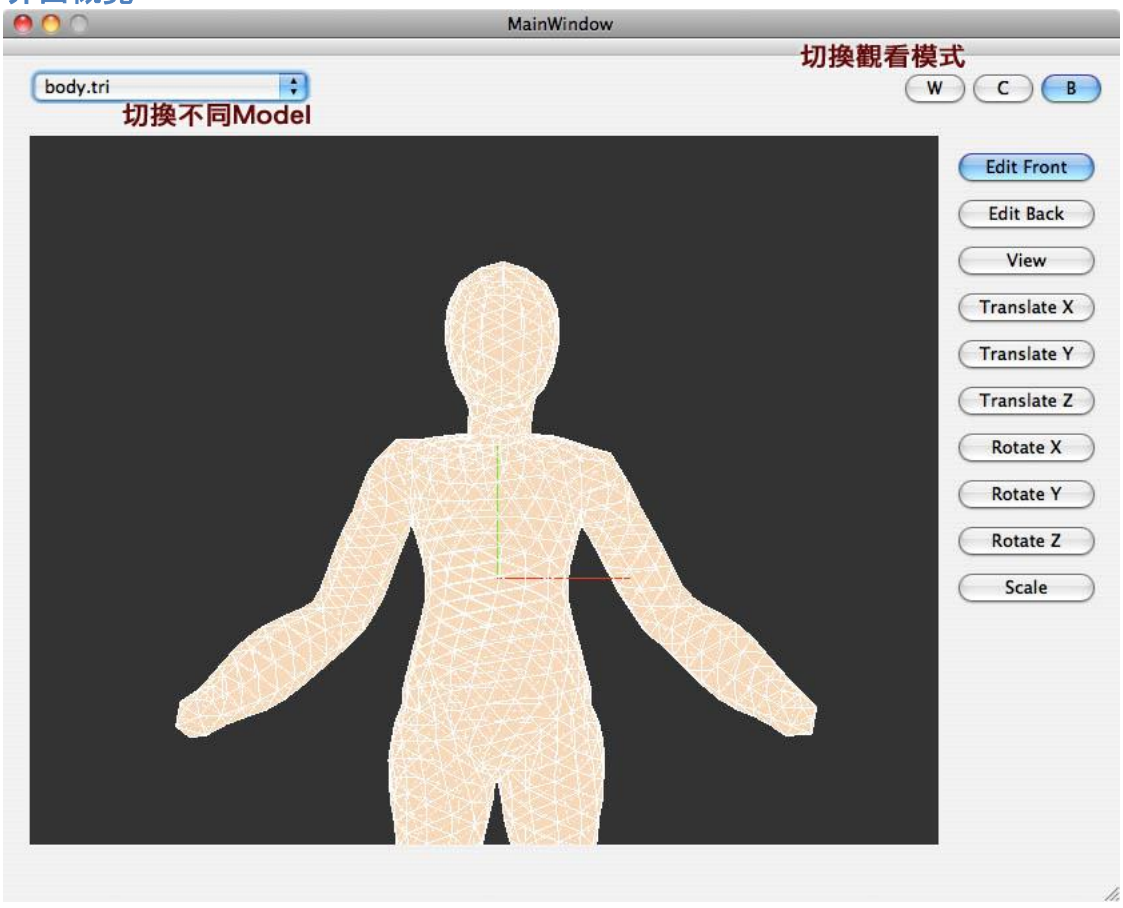
(e) 從 (c) ~ (d)中記錄下的 z 值中，找出最靠近出發點（near z-clipping plane）的 z 值回傳。

同樣的，我們對物體背面的部份，也是依照上述之方法，找出相對應之 z 值。（但由 far z-clipping plane 出發，到 near z-clipping plane。）

最後，我們對 inside、boundary pixel 所求出的 z 值，依照其鄰近 inside、boundary pixel 之貢獻做 smoothing。再連結 inside、boundary pixel 成為一個個三角形，並將 boundary pixel 移動到適當的輪廓線位置上，來完成一個符合使用者設計圖輪廓的 3D 立體模型。

五、操作方式

(1) 介面概覽

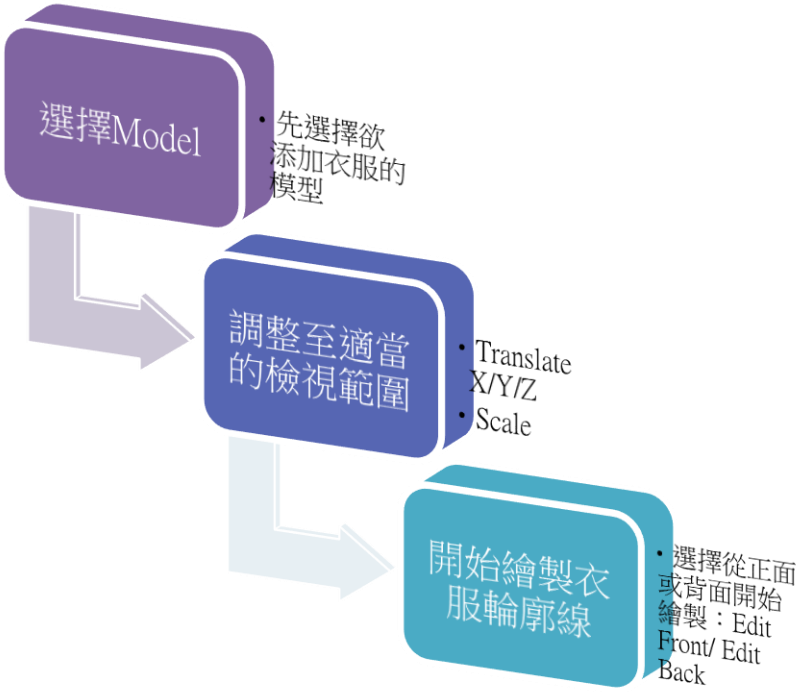


— 左上角選單：可以切換不同欲添加衣服的模型

— 右上角三個單選選單按鈕（W、C、B）：切換觀看 Model 的模式，W: Wireframe;　C: Colored;　B: Wireframe + Colored

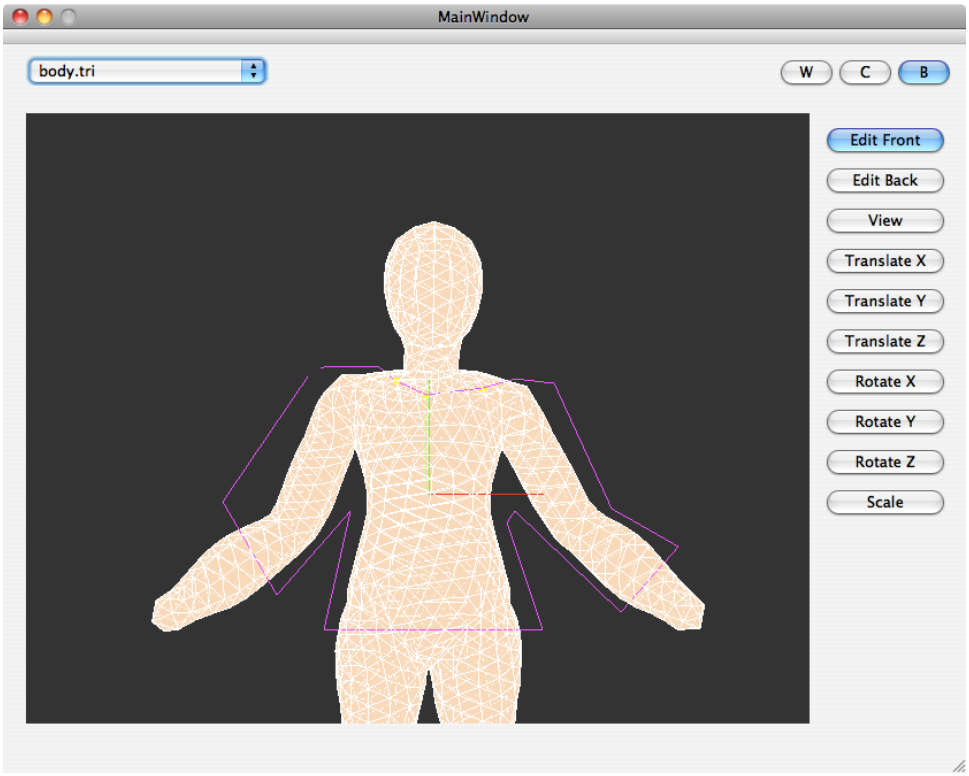
— 右側十個單選按鈕則讓設計者可以以多元的角度檢視 Model，然後再添加衣服

(2) 操作流程



(3) 繪製衣服輪廓線

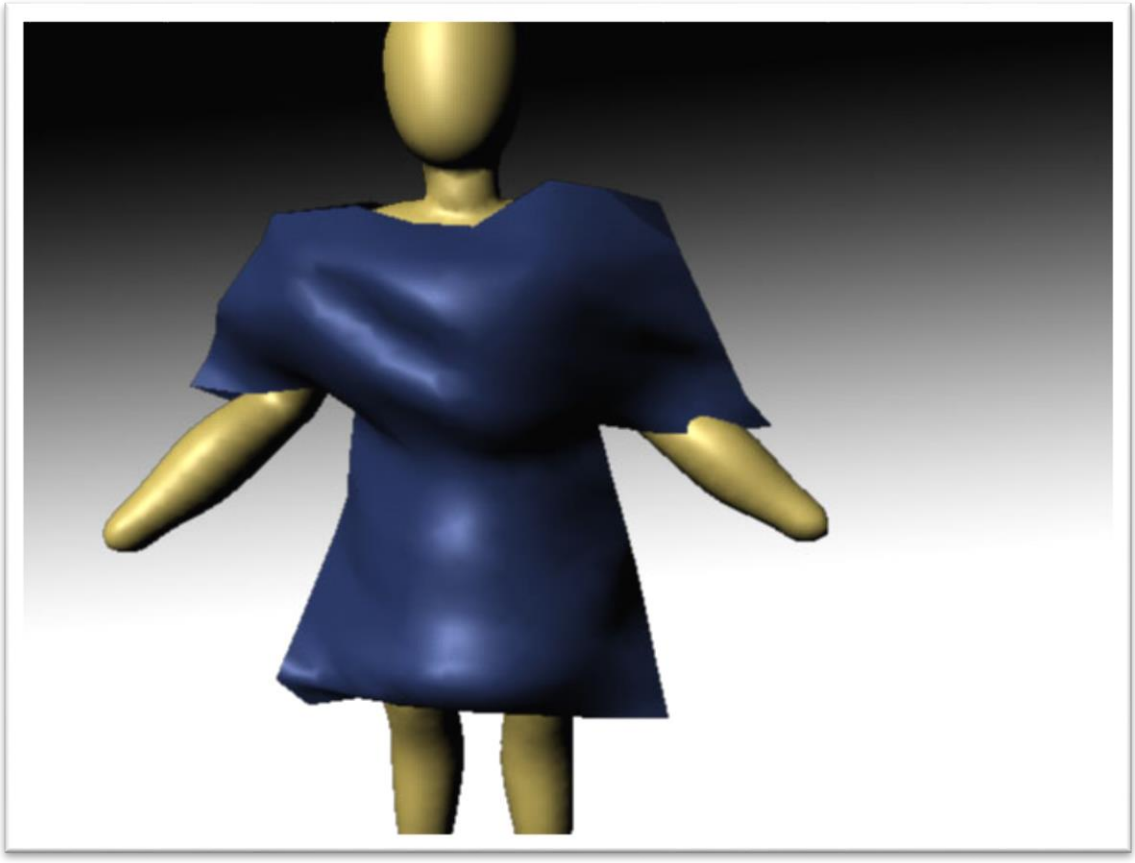
觀看模式先切換至 B（Colored + Wireframe）/ C (Colored)，接著在右側按鈕列點選 Edit Front（從 Model 正面繪製）/ Edit Back（從 Model 被面繪製），便可開始繪製衣服輪廓線。



接著在黑色有 Model 的區域點選左鍵即可開始進行繪製，每點擊一次滑鼠系統會拉一條線段，繪製完成後點選右鍵系統將自動進行 3D 衣服的運算，運算需要一些時間，系統計算完成後會將成果輸出在螢幕上，並自動生成 Model 穿上衣服後的.tri 檔案於執行目錄下。

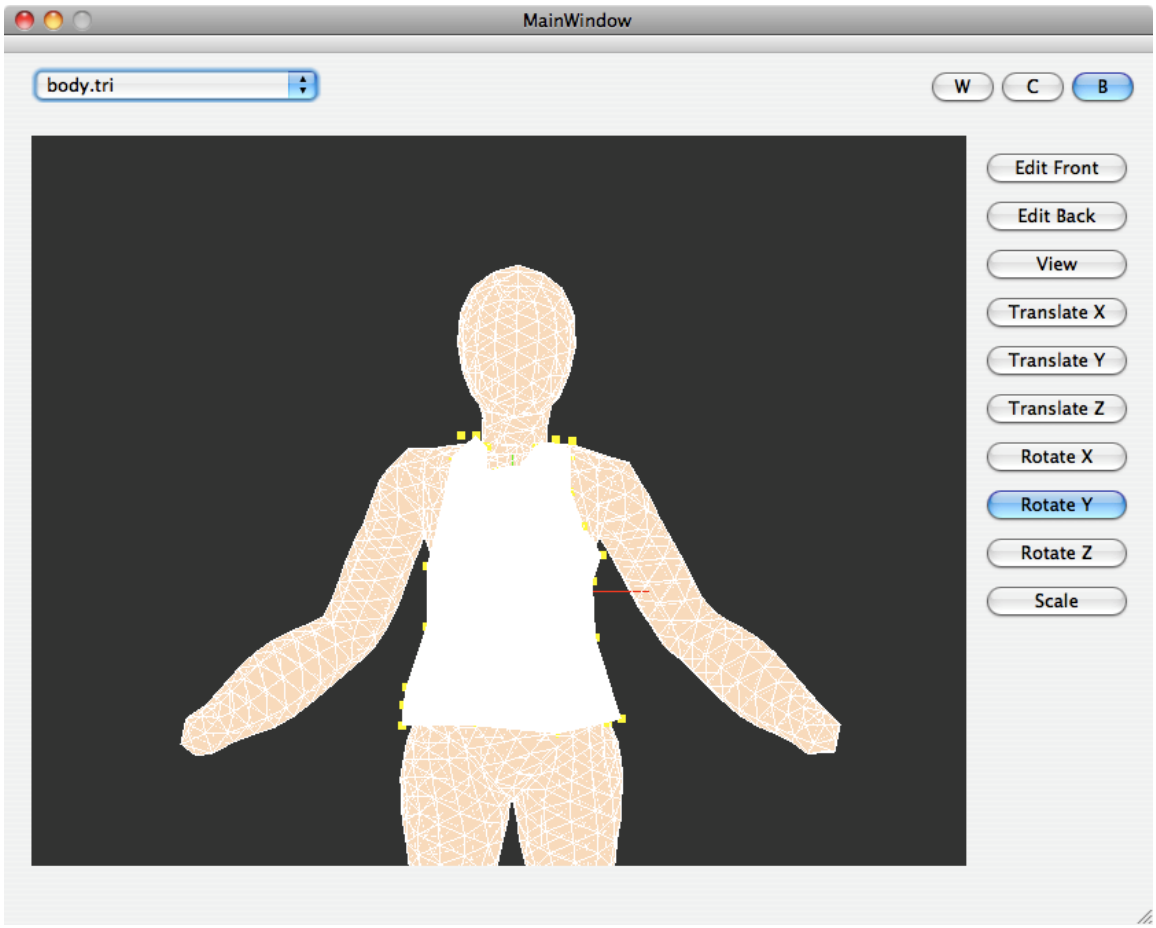
六、成果

成果一：

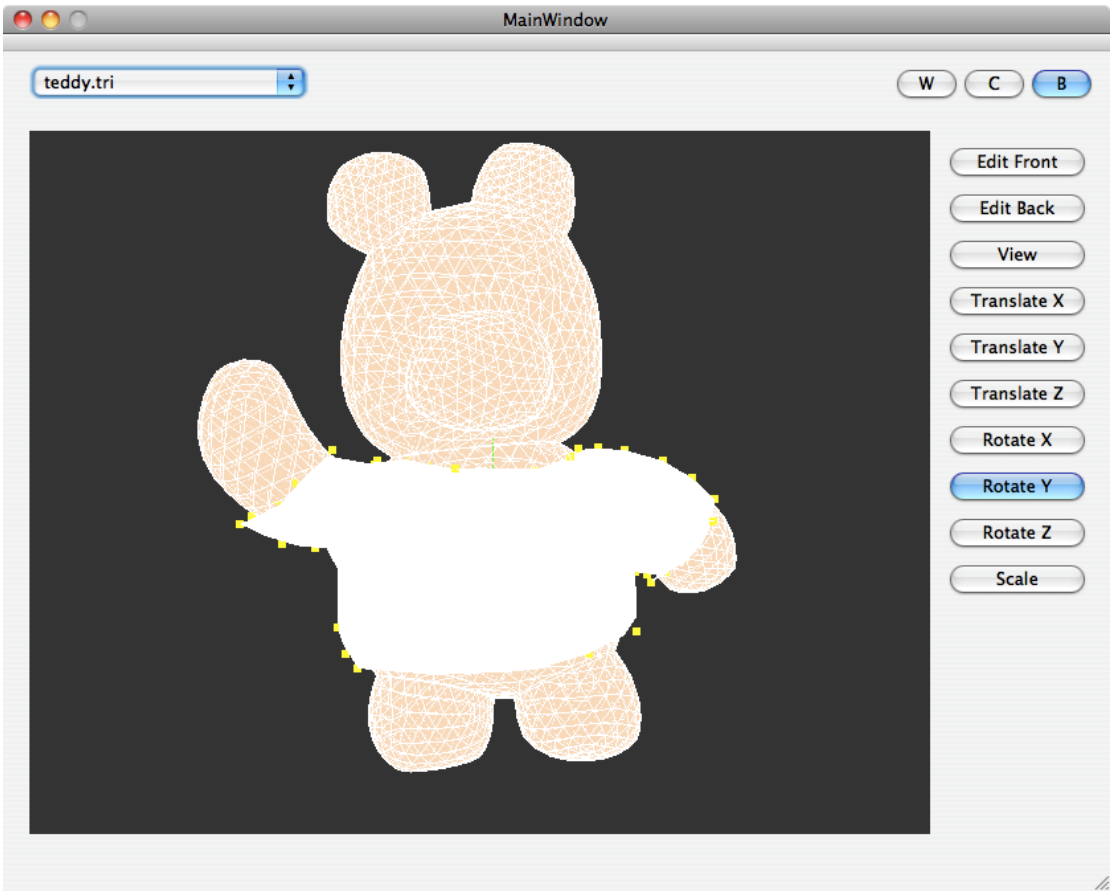


（經 Maya2010 上材質與打光）

成果二（上方為軟體運算結果，下方為 Maya 上材質與燈光後的結果）：



成果三：


MainWindow
teddy.tri
W
C
B
Edit Front
Edit Back
View
Translate X
Translate Y
Translate Z
Rotate X
Rotate Y
Rotate Z
Scale