

國立虎尾科技大學

機械設計工程系

產品協同設計第三組

手足球

Table Football

學生：

設計二甲 40623105 陳微云

設計二甲 40623109 李如芳

設計二甲 40623112 王柏翔

設計二甲 40623118 楊秉澤

設計二甲 40623125 鍾旻諺

設計二甲 40623127 張育偉

設計二甲 40623128 張華倞

設計二甲 40623135 洪明棋

設計二甲 40623141 何立翔

設計二甲 40623156 林聖翰

指導教授：嚴家銘

摘要

手足球系統設計

手足球系統模擬

送球機構設計

送球機構模擬

手足球系統功能

目錄

摘要	i
目錄	ii
表目錄	iii
圖目錄	iv
第一章 前言	1
第二章 設計與繪圖	2
2.1 零組件尺寸分析	2
2.2 參數設計與繪圖	4
2.3 參數設計與繪圖	9
2.4 細部設計與 BOM	9
第三章 送球機構設計與模擬	22
3.1 送球機構設計	22
3.2 送球機構模擬	22
第四章 手足球系統模擬	23
4.1 人物簡化	23
4.2 軸、篩子簡化	30
4.3 足球桌簡化	31
4.4 模擬步驟	34
第五章 系統功能展示	44
5.1 雙人鍵盤控制對打	44
5.2 雙電腦對打	44
5.3 單人鍵盤控制與電腦對打	45
5.4 影像辨識	46
5.5 影像辨識-canny	48
5.6 影像辨識電腦對打	49
第六章 參考文獻	51

表目錄

圖目錄

圖 x.1	送球軌道草圖	2
圖 2.1	撥桿修改圖	2
圖 2.2	送球機構本體修改圖	3
圖 2.3	入球口修改圖	3
圖 x.2	國際球桌	4
圖 2.4	球桌	5
圖 2.5	球員	6
圖 2.6	送球機構零件一	7
圖 2.7	送球機構零件二	7
圖 2.8	球桌組合圖	8
圖 2.9	送球機構組合圖	8
圖 2.10	球員尺寸更改圖	10
圖 2.11	球門 3D 圖	11
圖 2.12	發球機構本體	12
圖 2.13	撥桿	13
圖 2.14	球桌	14
圖 2.15	手把	15
圖 2.16	桿子	16
圖 2.17	球門三視圖	17
圖 2.18	球員工程圖	18
圖 2.19	塞子工程圖	19
圖 2.20	送球軌道工程圖	20
圖 2.21	爆炸圖	21
圖 x.3	送球機構設計	22

圖 4.1	onshape 匯出	23
圖 4.2	onshape 匯出對話框	24
圖 4.3	匯入.STL 檔後的對話框	24
圖 4.4	分離模型步驟點選	25
圖 4.5	模型爆開後	25
圖 4.6	頁面選擇器工具欄按鈕	25
圖 4.7	形狀編輯模式工具欄按鈕	25
圖 4.8	框選人物頭部	25
圖 4.9	簡化的對話框 - 人物	26
圖 4.10	頭部簡化完成後	26
圖 4.11	框選人物頭部以下的部位	27
圖 4.12	頭部簡化完成後	27
圖 4.13	合併物件	28
圖 4.14	新物件 people_dyn 新加並貼回原位	28
圖 4.15	合併物件	28
圖 4.16	使用 ctrl 在點擊工具欄按鈕	28
圖 4.17	物件傳送工具欄按鈕	29
圖 4.18	工具欄按鈕對話框	29
圖 4.19	貼完後	29
圖 4.20	框選軸需簡化的部分	30
圖 4.21	簡化對話框 - 軸	30
圖 4.22	軸簡化後產生的新圓柱	31
圖 4.23	框選篩子需簡化部分	31
圖 4.24	篩子簡化後產生的新物件	32
圖 4.25	選取三角形後按 Extract cuboid	32
圖 4.26	OK	33
圖 4.27	關閉 Shape Edition 後按 NO	33
圖 4.28	View/modify geometry	34
圖 4.29	取消勾選並調整板厚	35
圖 4.30	X 方向調整	35
圖 4.31	調整位置	36

圖 4.32 多選取板厚的三角形	36
圖 4.33 兩種方式厚度比較	36
圖 4.34 簡化後的板子相互干涉	36
圖 4.35 將所需物件拉入 Table_dyn 下-1	37
圖 4.36 將所需物件拉入 Table_dyn 下-2	37
圖 4.37 平移軸和軸相貼合	38
圖 4.38 物件旋轉工具欄按鈕	38
圖 4.39 平移軸和軸方向相同	38
圖 4.40 平移軸、旋轉軸及軸位置及方向皆一致	39
圖 4.41 樹狀圖排列	39
圖 4.42 合併並排列好	39
圖 4.43 Scene Object Properties 對話框	40
圖 4.44 對話框勾選 - Table_dyn	41
圖 4.45 對話框勾選 - cylinder-L1-dyn	41
圖 4.46 對話框勾選 - s-L1-2-dyn	42
圖 4.47 對話框勾選 - Prismatic-joint-1	42
圖 4.48 對話框勾選 - Revolute-joint-1	43
圖 4.49 模擬圖	43
 圖 5.1 玩家對打	44
圖 5.2 電腦對打	45
圖 5.3 玩家電腦	46
圖 5.4 原圖	47
圖 5.5 模糊處理後	47
圖 5.6 hsv 處理後	48
圖 5.7 高斯濾波	48
圖 5.8 尋找 Gx 與 Gy	49
圖 5.9 找到強度斜率與方向	49
圖 5.10 守門員程式架構	50

第一章 前言

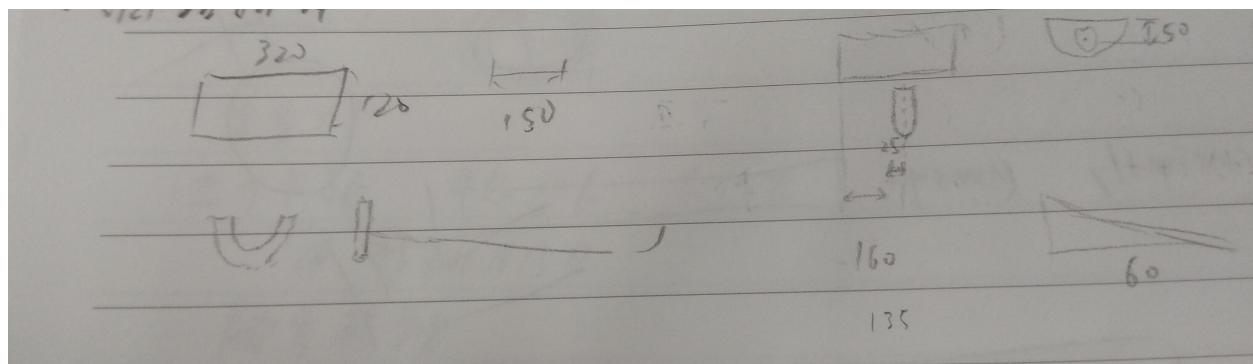
由於每個人都有各自擅長的領域，因此需要分工合作完成一個商品，每個人只需要往自己擅長的領域鑽入，可以不用花費大量時間在自己所不擅長的領域。因此藉由協同製作手足球系統來更加了解協同時會遇到的問題，並了解協同的重要性跟好處。

第二章 設計與繪圖

2.1 零組件尺寸分析

6. 送球軌道:

由於尺寸過小的關係，再加上需要配合送球機構以及軌道的關係修改了相當多次，為配合到機構之間的距離將尺寸從 120 改成 135



送球軌道草圖

7. 送球機構:

在撥桿的部分原本是呈現長方形的，因考慮到了沒有角度在撥球時可能會造成球的動向無法預期，所以在兩邊斜面的部分增加了 10 度的傾角。

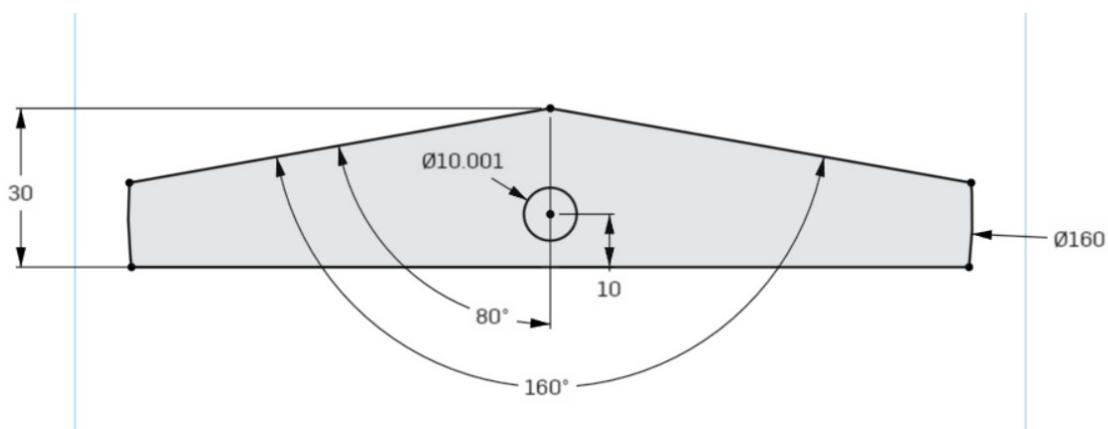


圖 2.1: 撥桿修改圖

搭配撥桿的造型後，在外殼的設計上各增加了厚度 5mm，而在與撥桿相差 2mm 的誤差是因為組員反映出在 v-rep 模擬時兩機件會互相干涉，所以更動了原先的尺寸。

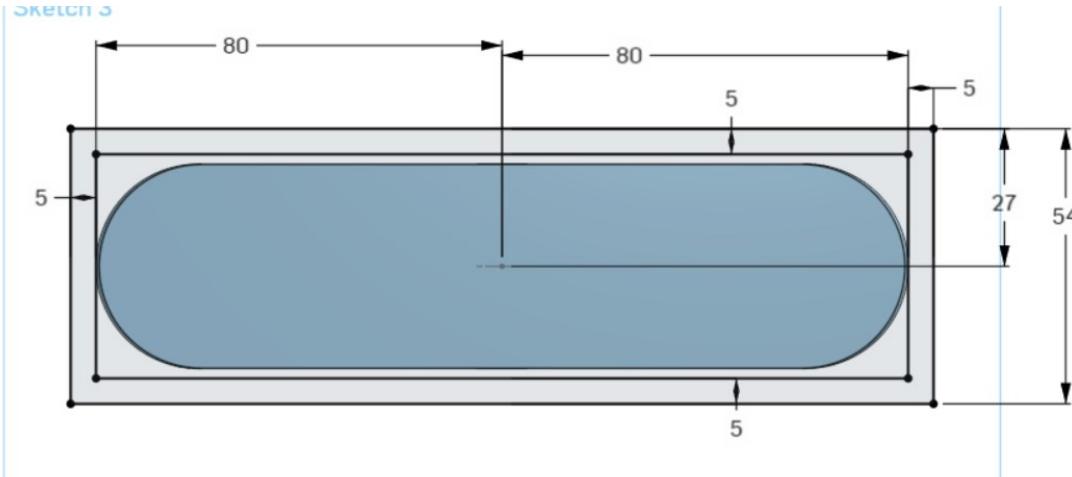


圖 2.2: 送球機構本體修改圖

在進球口高度的設計上，為了方便負責軌道組員的製作，所以將孔的位置標於圓的底部，而球口的高度也有經過設計，與撥桿的部分切齊在進球時不會造成干涉。

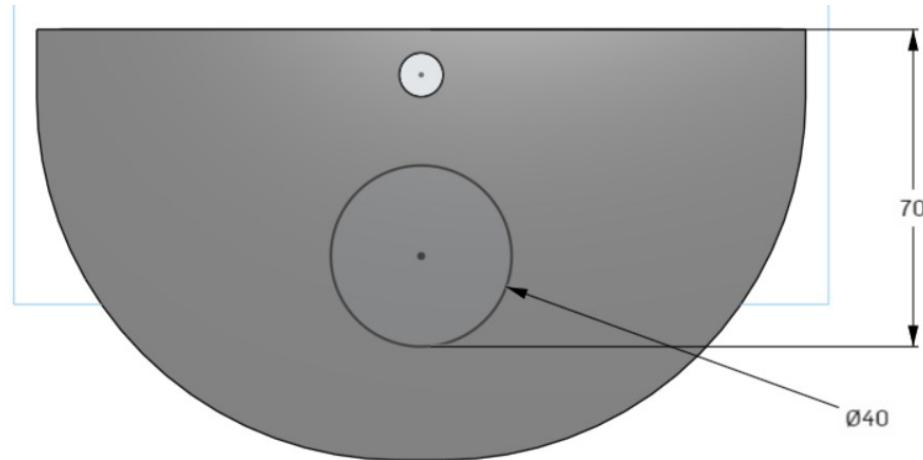


圖 2.3: 入球口修改圖

繪圖影片

<https://youtu.be/l5HXjuSZNig>

2.2 參數設計與繪圖

123 w11 進度

本周進行人員的工作分配，王柏翔與鍾旻諺分配到參數設計與繪圖的工作，我們先上網搜尋手足球桌的國際規格之後，參考國際比賽桌的尺寸之後再做了一些些微的調整，為了之後設計和配合可以方便一點。



國際球桌

W12 進度

本周繪製完(球桌)(球員)，和一些((手把)和串聯球員的(桿件)，以及另外一個分

工小組的分球機構還在等我們的尺寸確定，他們的送球軌道尺寸才能配合到我們的球桌上，配合之後我們兩邊再去做一些尺寸的微調，讓分球機構能完美的和球桌合體。

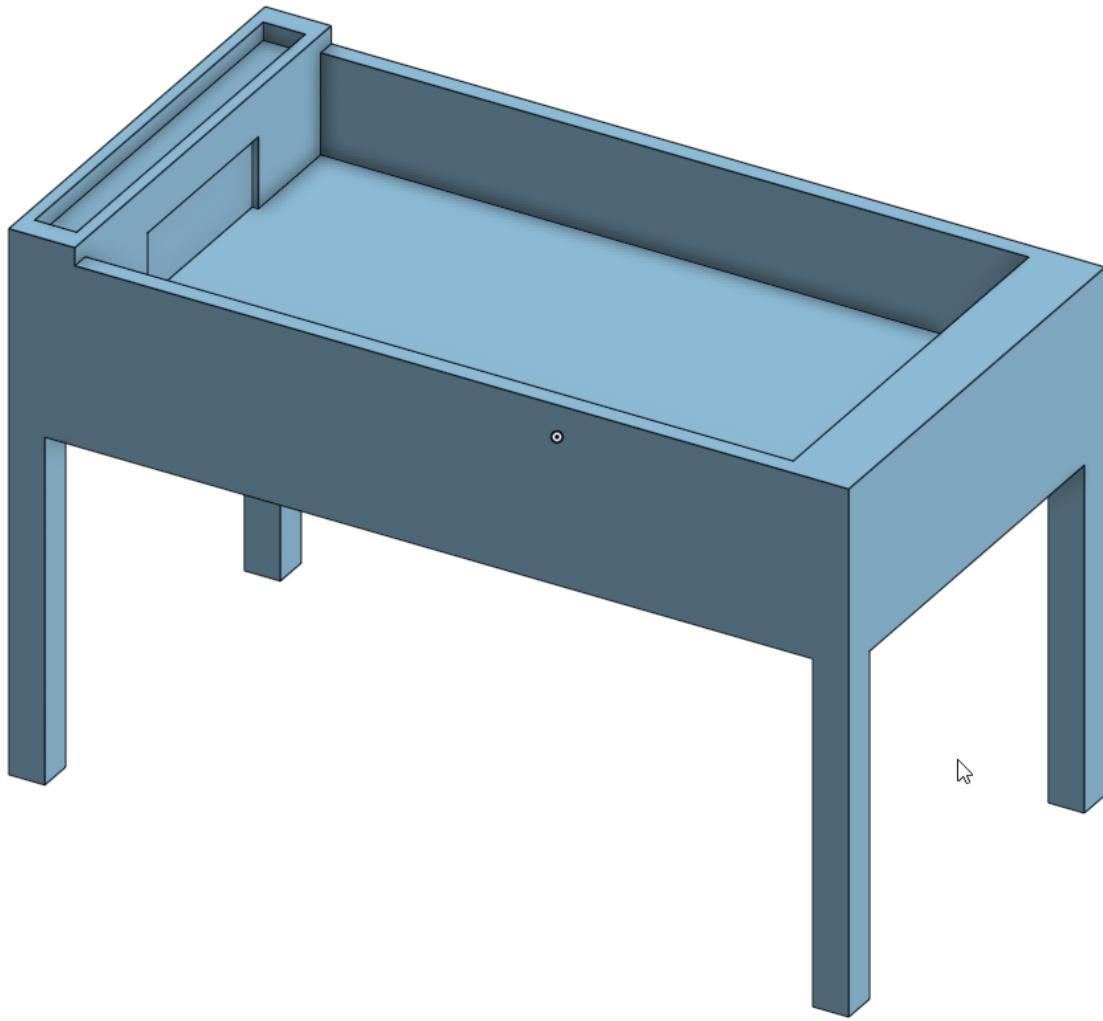


圖 2.4: 球桌

W13 進度

本周我們進行零組件、送球機構、足球桌的組裝，組裝之後我們發現各組件在配合上有些許的誤差，在設定球員和手把連桿時我們為了設定它的移動範圍花了許多的時間，因為不知道為什麼同樣的參數卻不能套用在同位置的球桿上，所以我們只好每根都自己慢慢的去試，才不會讓它過度移動(例如整根拔出來)，不過在其他組員分工的幫忙下順利的組裝完成了。

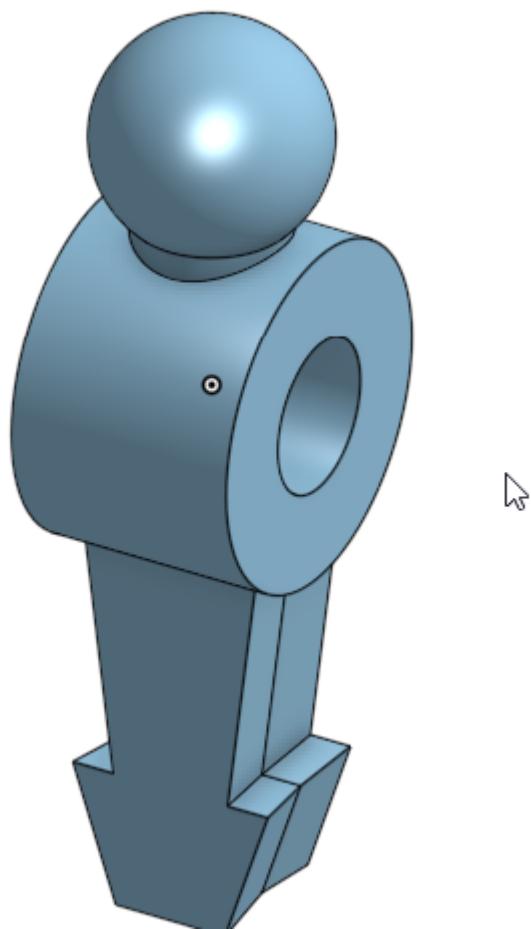


圖 2.5: 球員

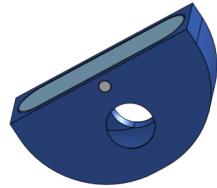


圖 2.6: 送球機構零件一

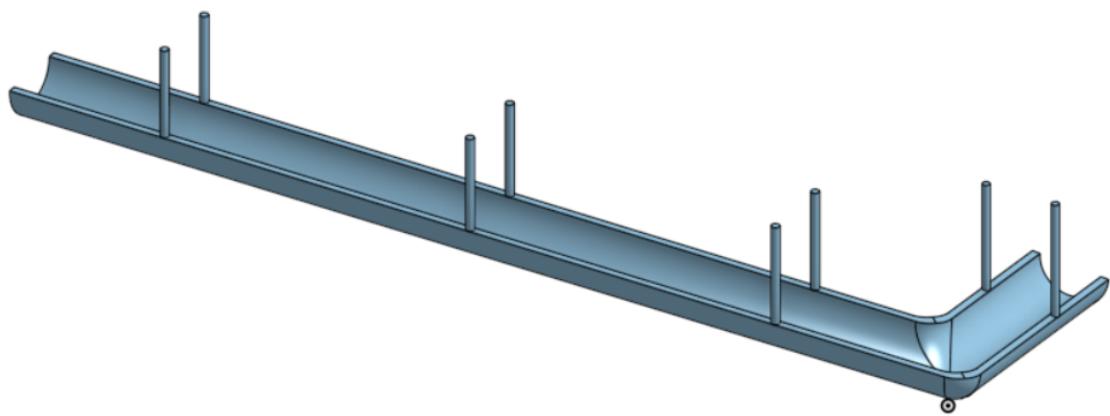


圖 2.7: 送球機構零件二

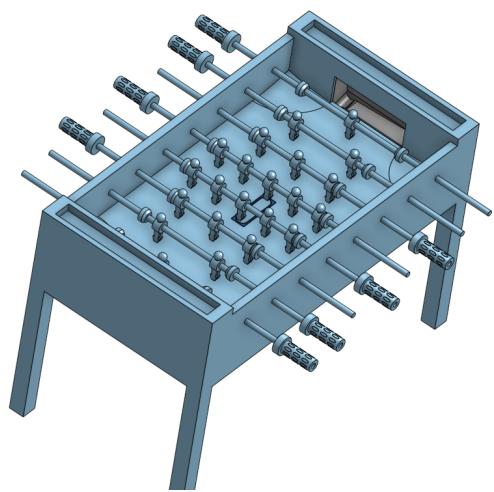


圖 2.8: 球桌組合圖

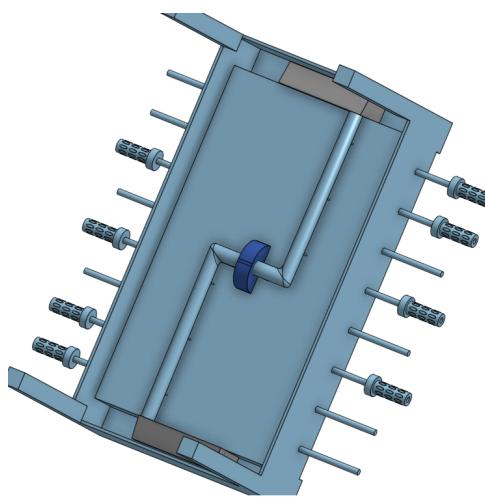


圖 2.9: 送球機構組合圖

W14 進度

我們把所有機構做組裝，發現球員與球員間會互相碰撞所以去做了尺寸上的更改，把球員長度改到以下尺寸並把球桌的洞上下間距做更改。

W15 進度

我們前兩個禮拜已經把所有機構做好限制，這禮拜做最後檢查然後就把圖交由操作 V-rep 組員去做簡化與模擬。

W16 進度將送球機構整個過程的零件進行修改，讓球可以順利的循環至指定地點，程式部分尚未完成所以無法達到預期經由 sensor 偵測後把球送給指定方。將所有零件資料與零件 bom 圖從 onshape 整理出來。

8. 球門:

球門的外型依照球桌的大小與軌道的進球口進行尺寸配合，內部為了讓球可以順利的進入軌道進行了多次的斜度變動，在進球口處也利用圓角讓球可以更順暢的流通，外型用 Draft 分上下殼修改外型的斜度。

2.3 參數設計與繪圖

送球機構繪製:<https://youtu.be/l5HXjuSZNig>

2.4 細部設計與 BOM

發球機構:

1. 發球機構本體:

工程圖連結網址請點這:<https://cad.onshape.com/documents/7063242033d0934280d360d8/w/172d28e1bcc6647f37e823d3/e/cc2fcff733a6da56f285b8b8>

2. 撥桿:

工程圖連結網址請點這:<https://cad.onshape.com/documents/7063242033d0934280d360d8/w/172d28e1bcc6647f37e823d3/e/79a2471d3dbdab0247698c7b>

3. 球桌:

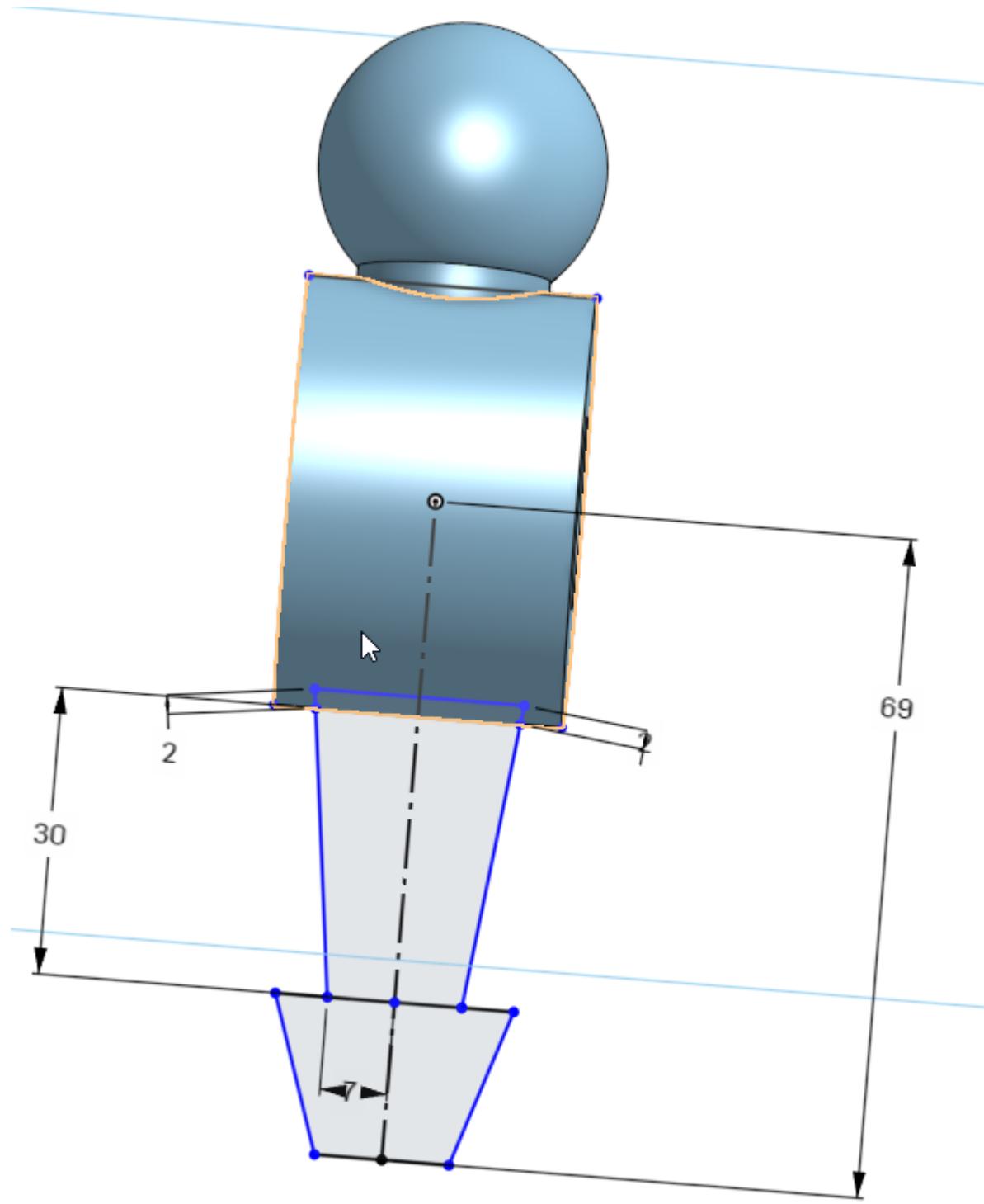


圖 2.10: 球員尺寸更改圖

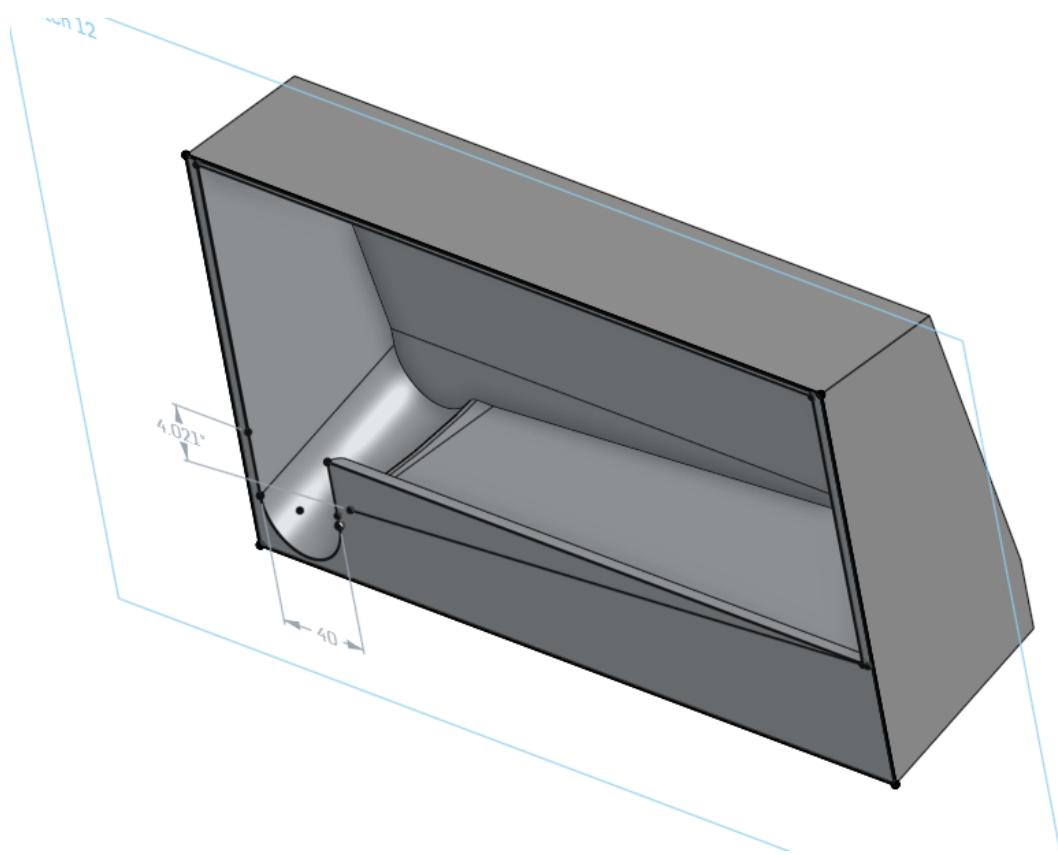


圖 2.11: 球門 3D 圖

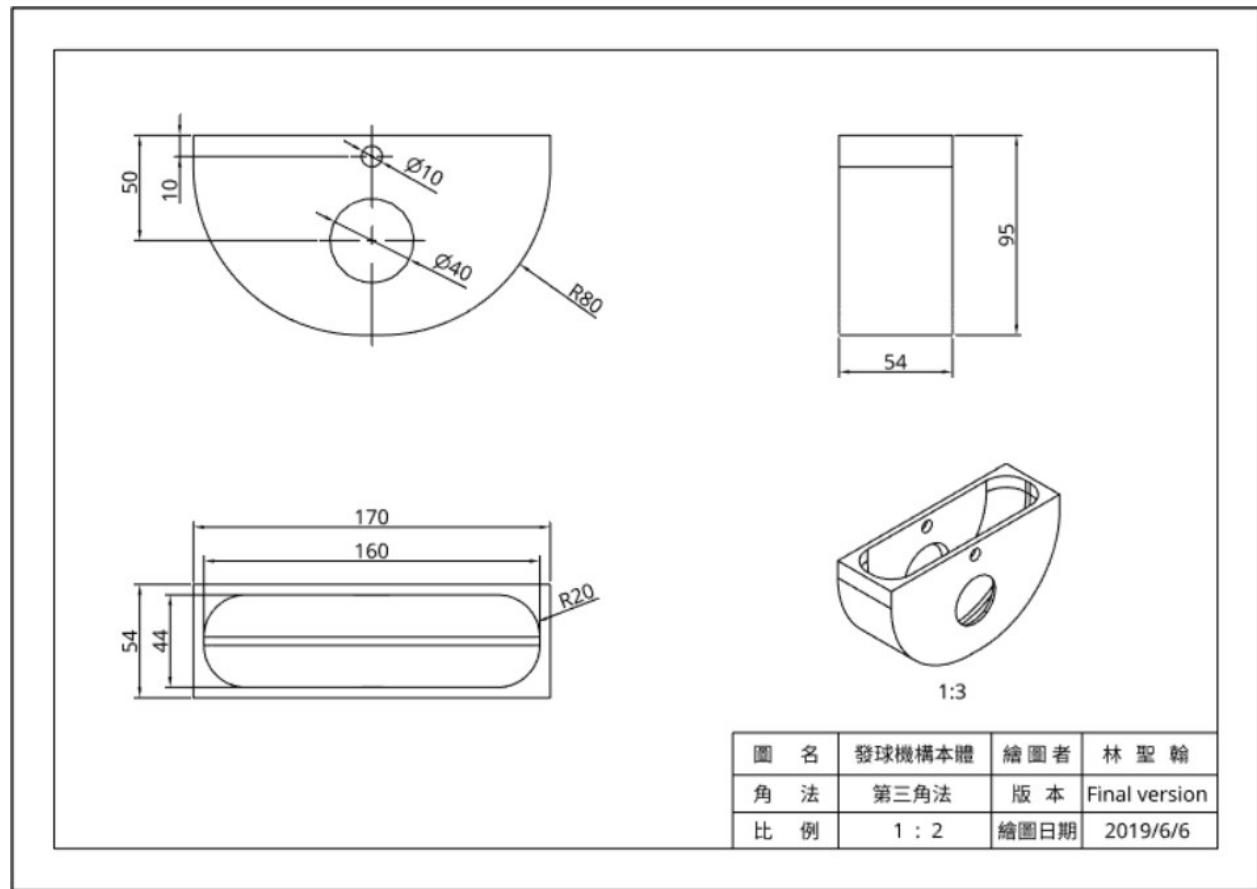


圖 2.12: 發球機構本體

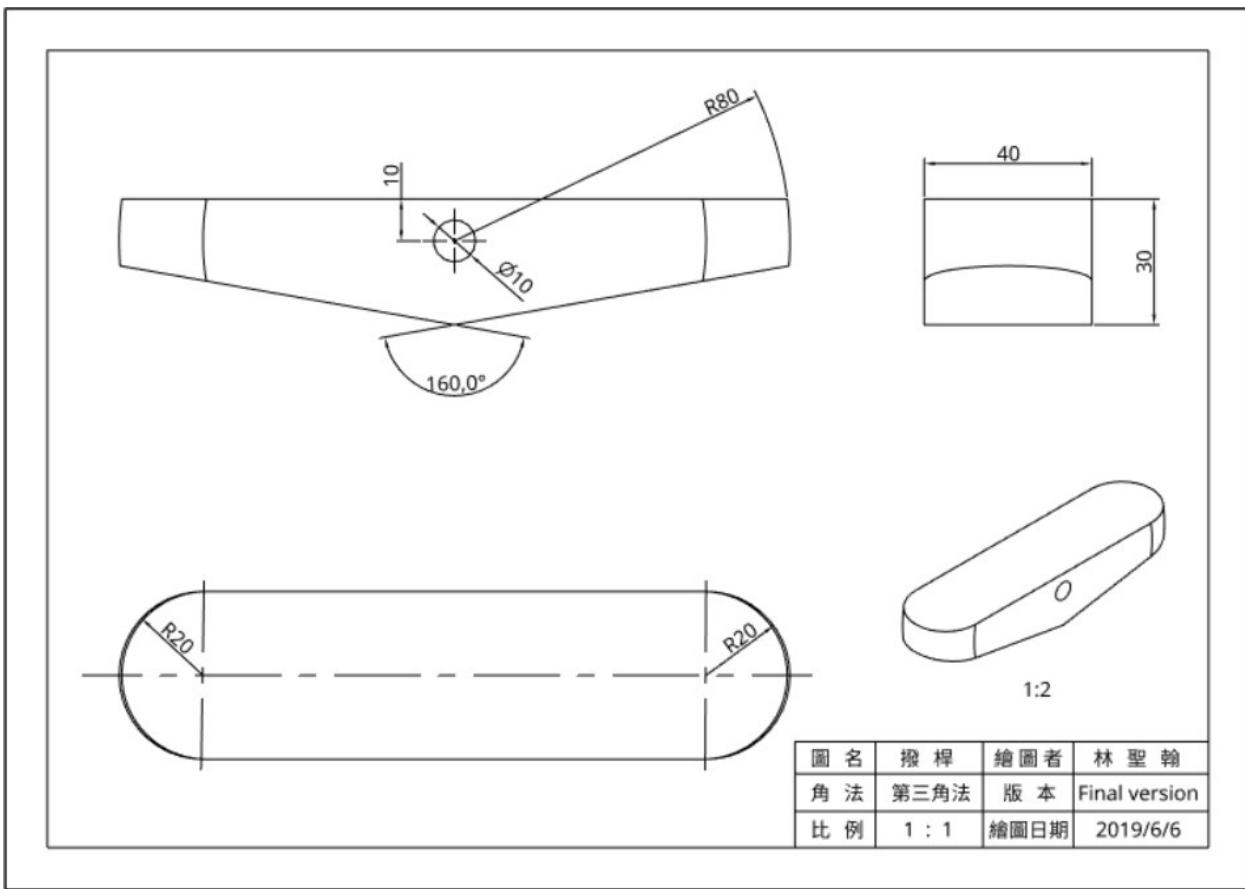


圖 2.13: 撥桿

工程圖連結網址請點這:<https://cad.onshape.com/documents/7063242033d0934280d360d8/w/172d28e1bcc6647f37e823d3/e/af2860e27a45e3d65e4744ae>

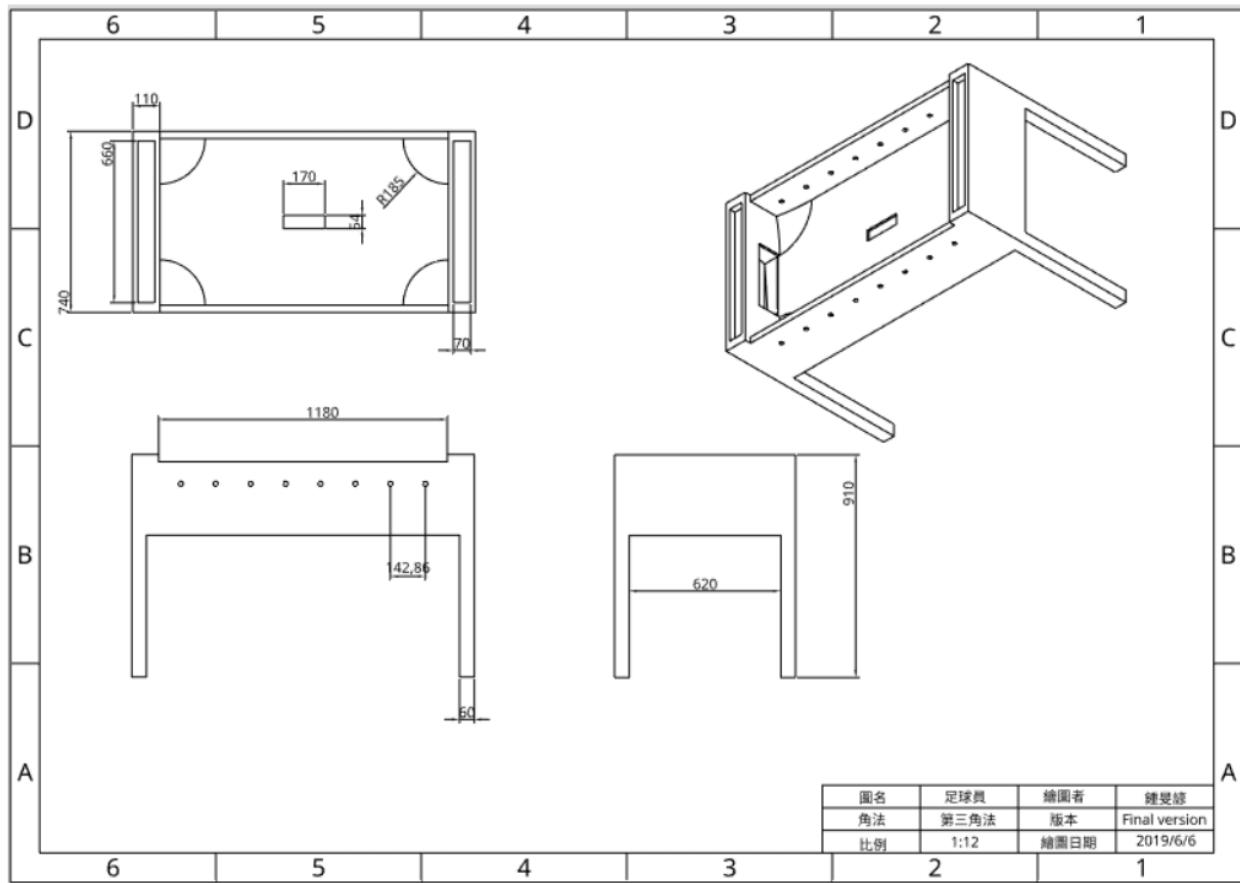


圖 2.14: 球桌

4. 手把:

工程圖連結網址請點這:<https://cad.onshape.com/documents/7063242033d0934280d360d8/w/172d28e1bcc6647f37e823d3/e/0ffbfb2d9caded49fe7b47679>

5. 桿子:

工程圖連結網址請點這:<https://cad.onshape.com/documents/7063242033d0934280d360d8/w/172d28e1bcc6647f37e823d3/e/558a2fa8dc9cd780958dc74f>

6. 球門:

工程圖連結網址請點這:<https://cad.onshape.com/documents/7063242033d0934280d360d8/w/172d28e1bcc6647f37e823d3/e/87427ddbdbb5063db4e9e7e7>

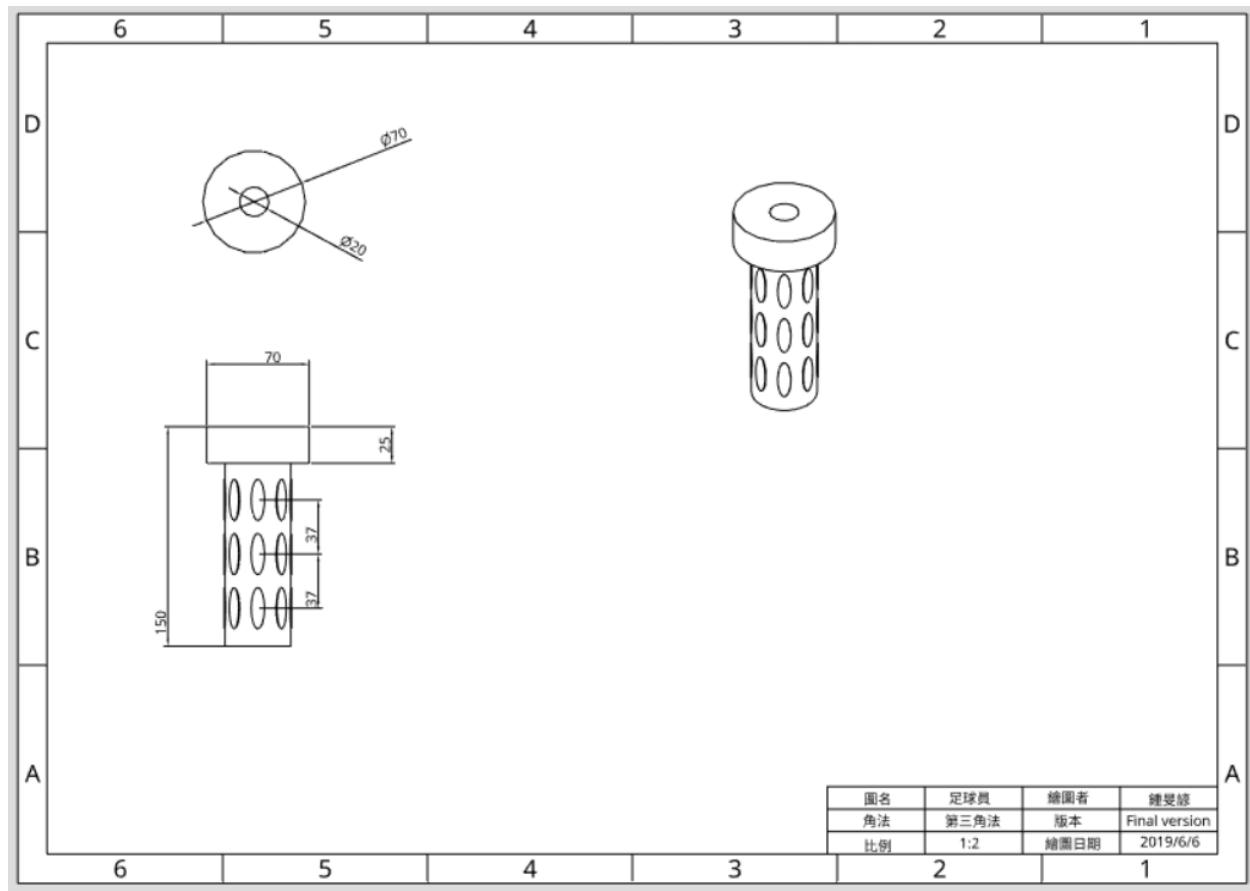


圖 2.15: 手把

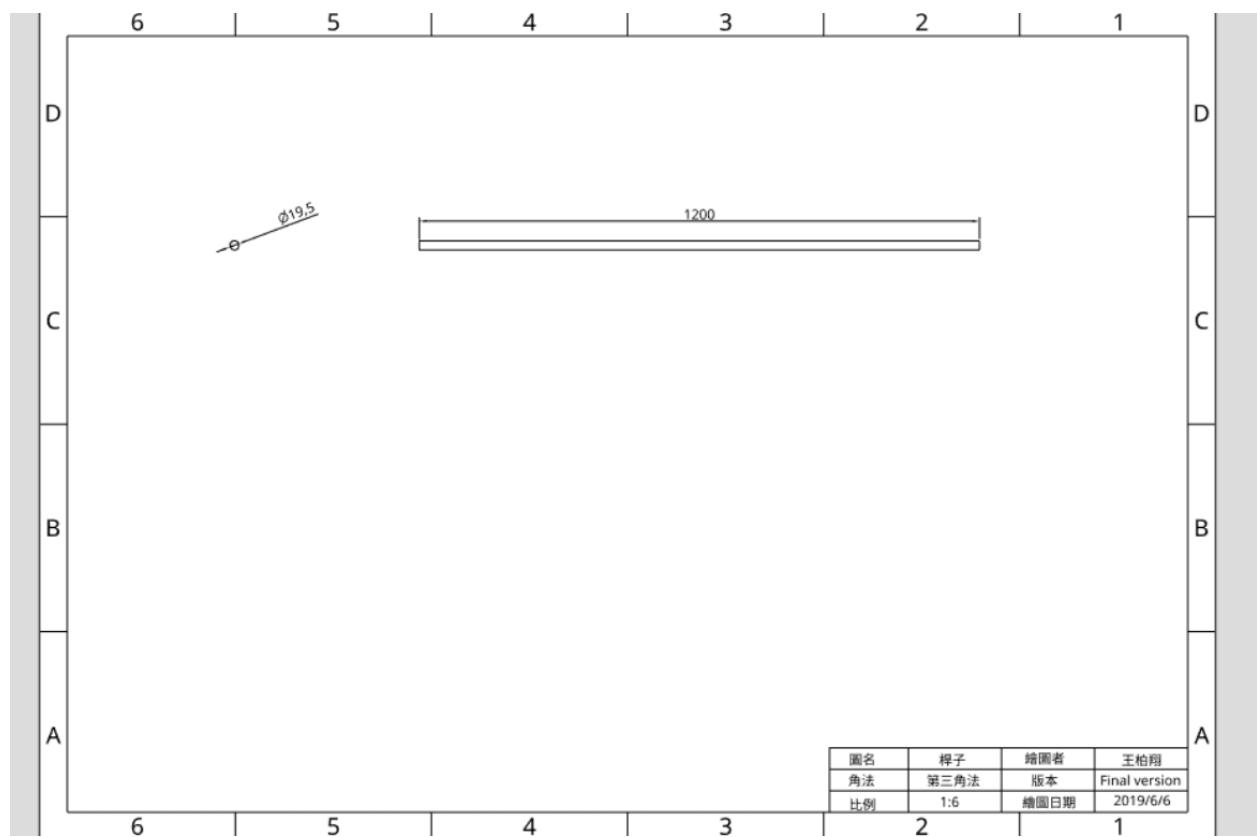


圖 2.16: 桿子

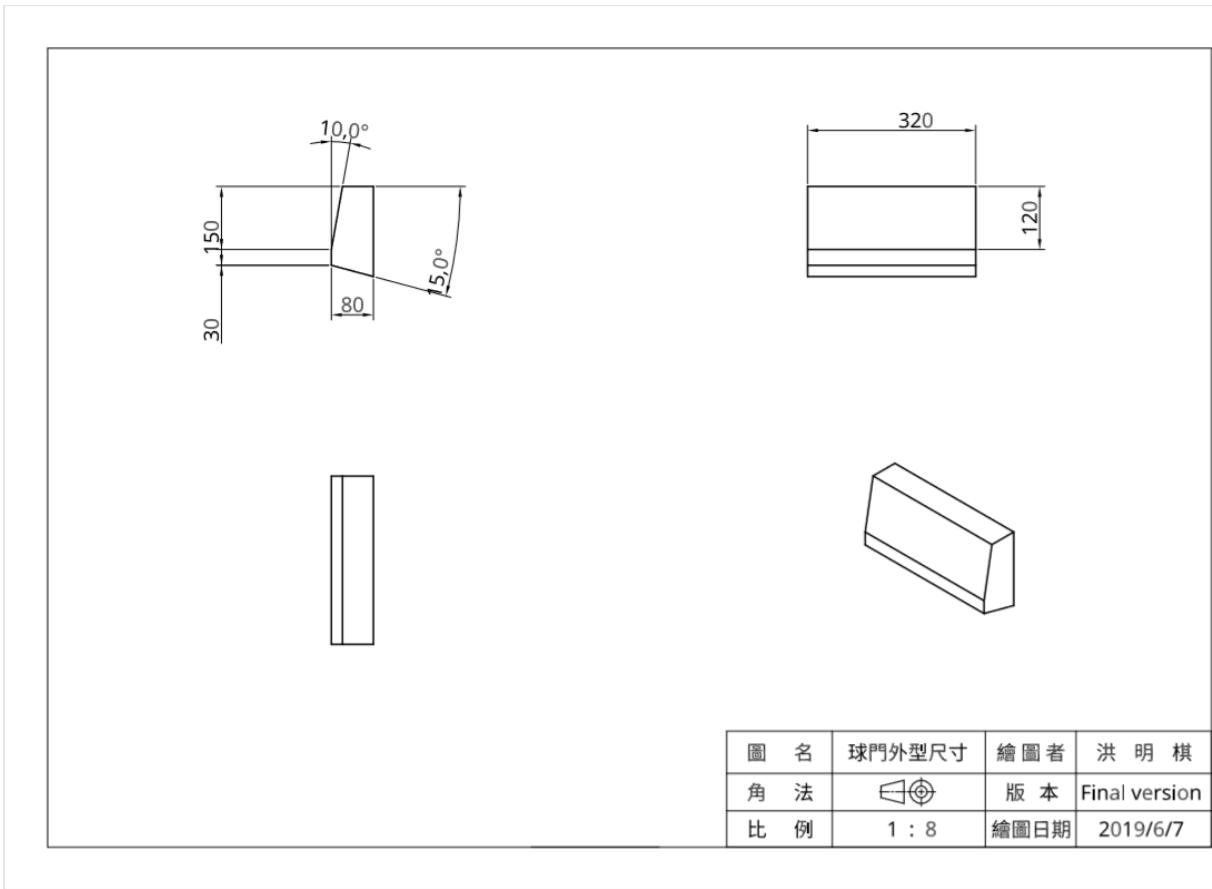


圖 2.17: 球門三視圖

7. 足球員工程圖連結請點這:<https://cad.onshape.com/documents/7063242033d0934280d360dw/172d28e1bcc6647f37e823d3/e/83f38694788c755ab69cb18c>

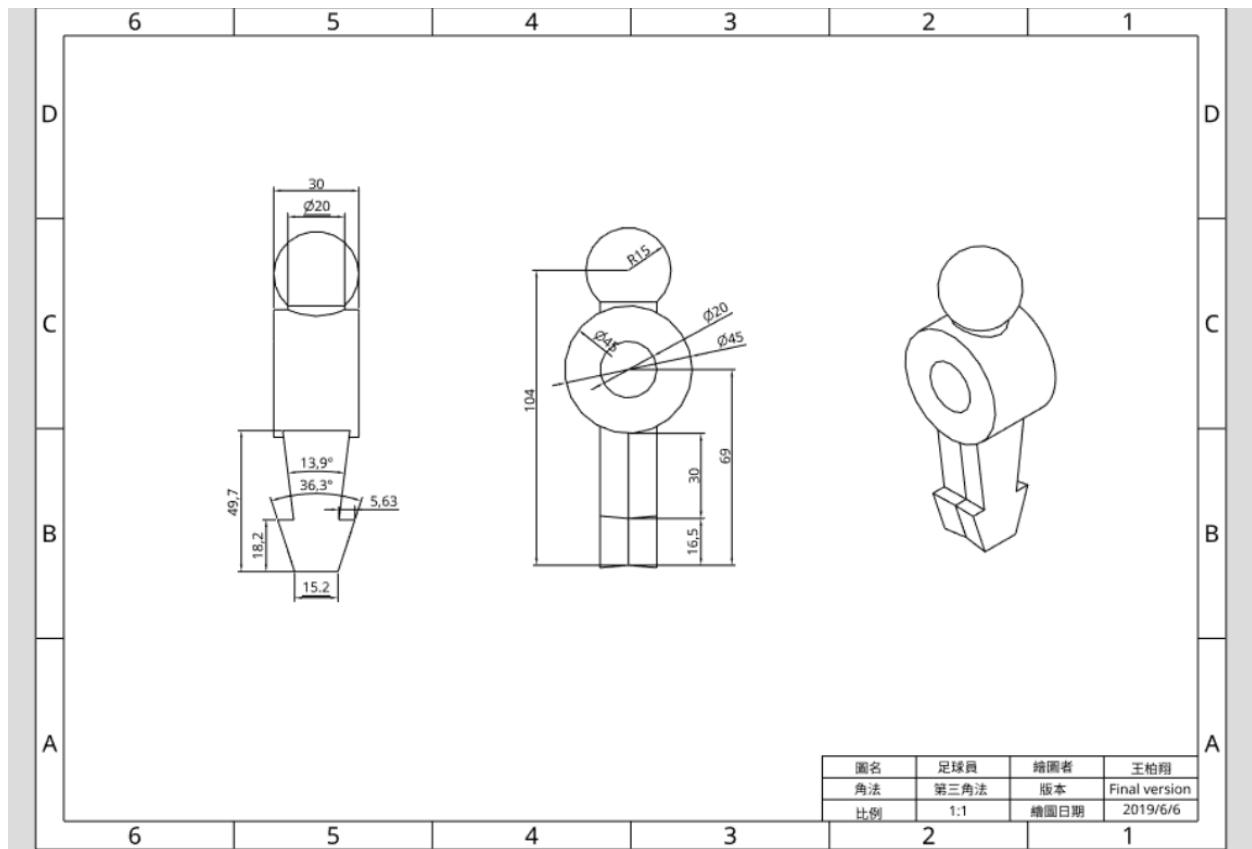


圖 2.18: 球員工程圖

8. 塞子工程圖連結請點這:<https://cad.onshape.com/documents/7063242033d0934280d360dw/172d28e1bcc6647f37e823d3/e/40911ab194ff8009ffdd6238>

9. 送球軌道

工程圖連結請點這:<https://cad.onshape.com/documents/7063242033d0934280d360dw/172d28e1bcc6647f37e823d3/e/09f93ed5107bb380078f0a91>

10. 爆炸圖連結請點這:<https://cad.onshape.com/documents/7063242033d0934280d360dw/172d28e1bcc6647f37e823d3/e/bf399e06fdf1d159d2657944>

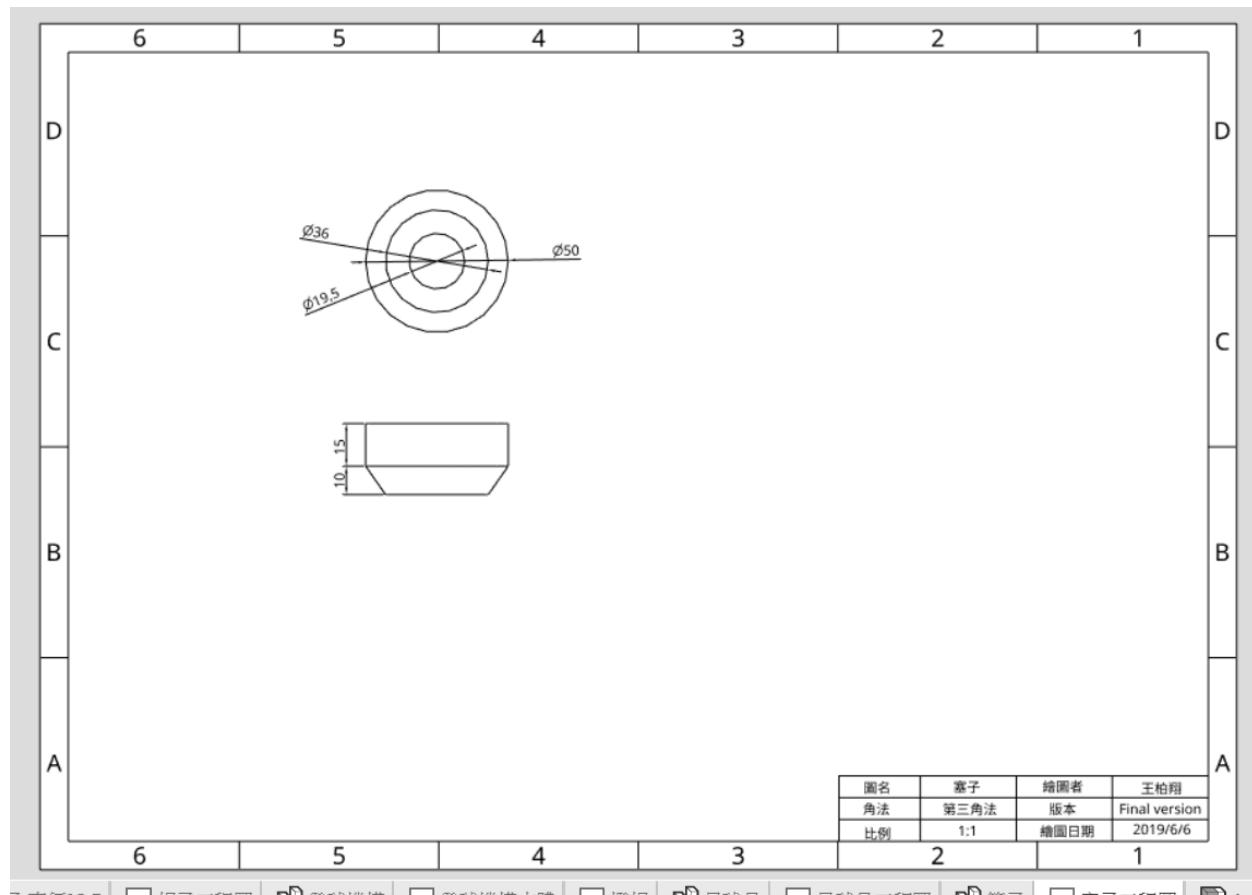


圖 2.19: 塞子工程圖

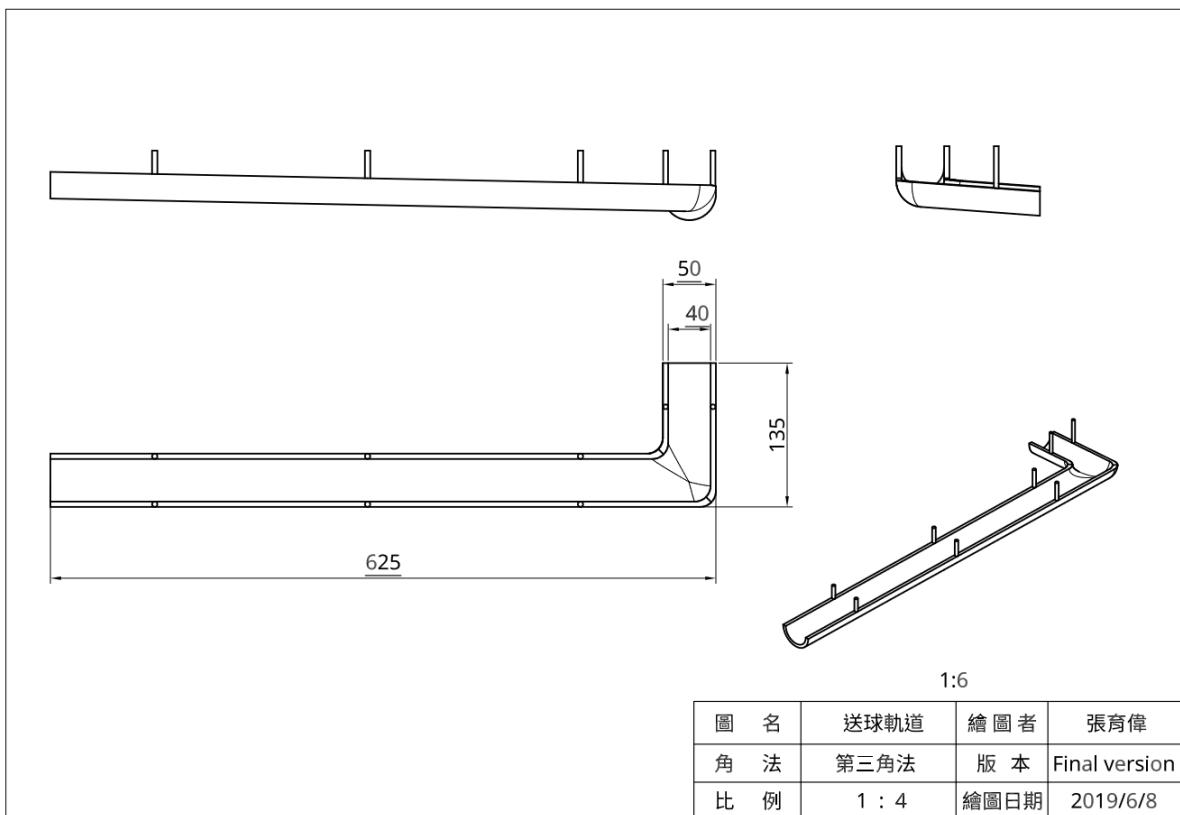


圖 2.20: 送球軌道工程圖

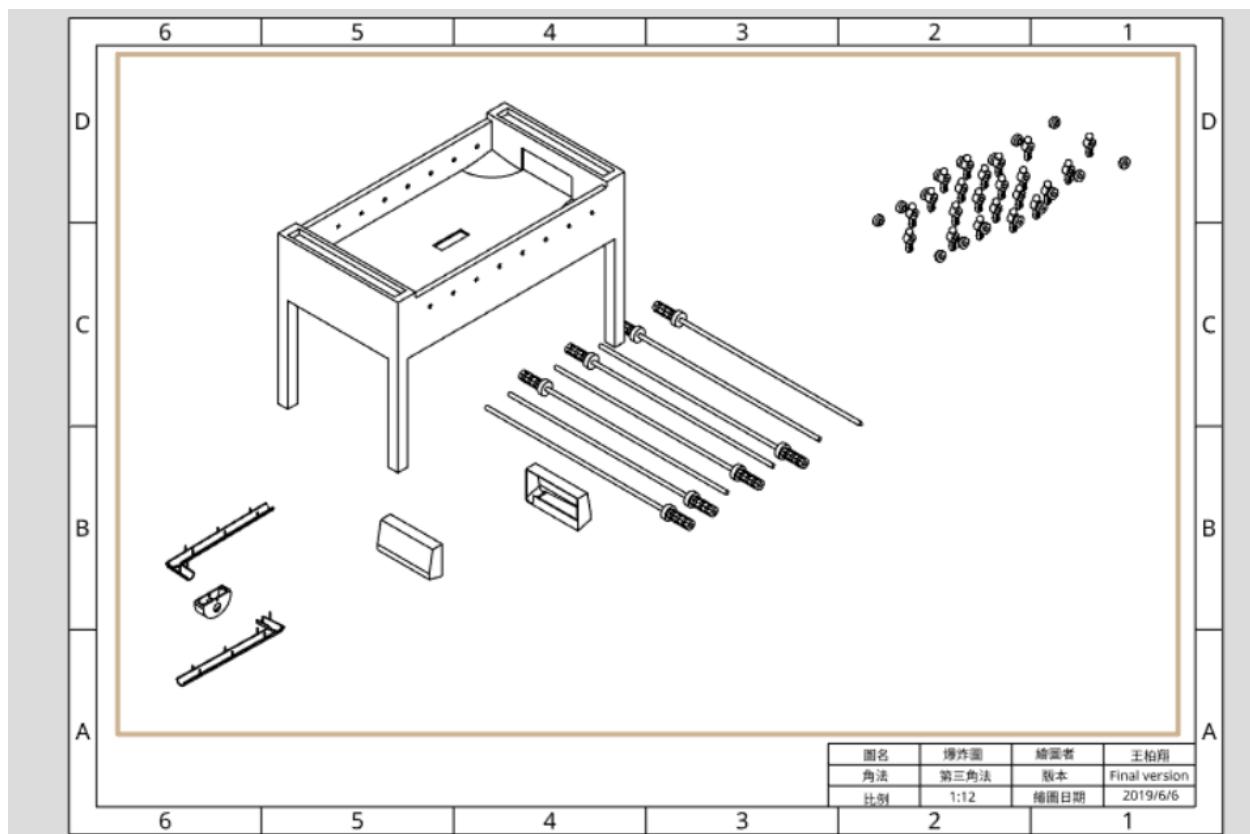
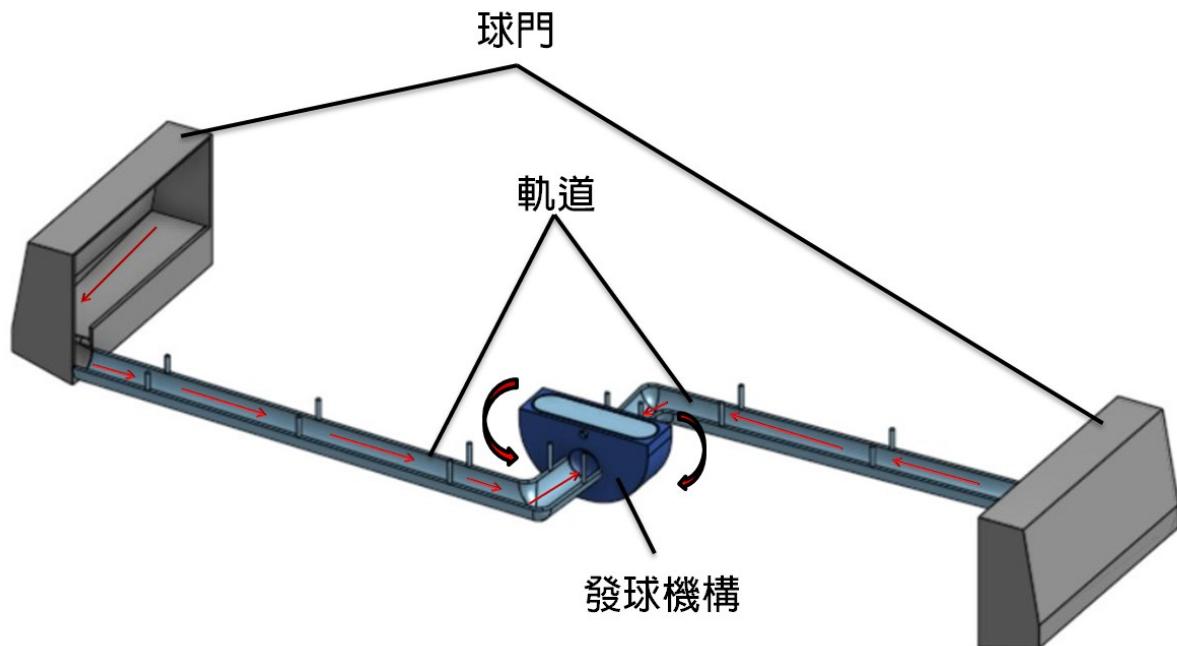


圖 2.21: 爆炸圖

第三章 送球機構設計與模擬

3.1 送球機構設計

本組將軌道的部分隱藏在球桌下方，軌道不凸出球桌外圍影響操作者操作，此外也將發球機構簡單化，利用單純的轉動系統讓球進入機構後以撥桿旋轉將球從球桌正中間發出，而選轉撥桿的好處還有能依得分系統的判斷，以正逆轉的方式控制球的動向。



送球機構設計

3.2 送球機構模擬

40623156

製作發球機構模擬，確定發球機構能順利進行發球的動作。

<https://youtu.be/VNxQo9RTC1Y>

第四章 手足球系統模擬

4.1 人物簡化

首先在 onshape 將想要使用的零件圖或組合圖按照圖 4.1 及圖 4.2 匯出成.stl 檔，再從 v-rep 中開啟.stl 檔 (使用 [File → Import → Mash...])，會出現如圖 4.3 的對話框，依據個人所需去做點選，在按 OK 即可在視窗中導入模型。可以從圖 4.4 中看出，導入的模型是未分離的模型 (如若是零件圖則不須此步驟)，所以我們使用 [Edit → Grouping/Merging → Divide selected shapes] 來將模型中的物件全都爆開，如圖 4.5。

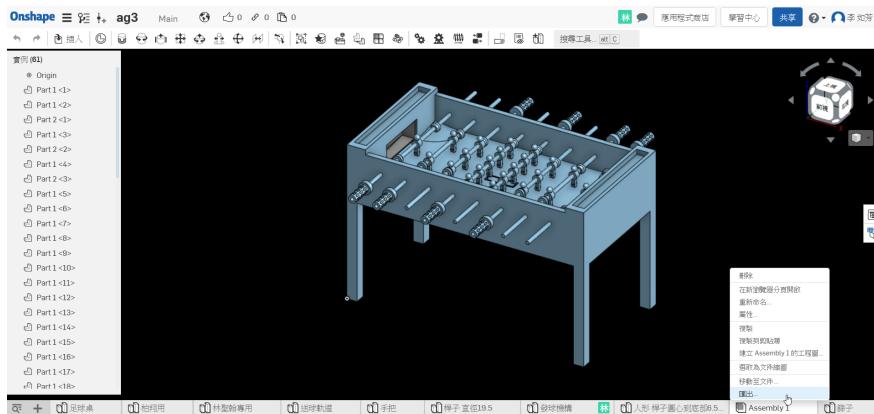


圖 4.1: onshape 匯出

接下來說明人物簡化步驟，先來進行人物頭部的簡化，先選擇人物將其複製 (使用 [Edit → Copy selected Objects]) 到一個新建的場景 (使用 [File → New scene])，再將人物貼上 (使用 [Edit → Paste buffer])。再點選頁面選擇器工具欄按鈕，如圖 4.5，使得在簡化的過程中更容易點選，接著選取人物再點選形狀編輯模式工具欄按鈕來進行簡化，如圖 4.6，在此我框選人物的頭部如圖 4.7，再點選簡化的對話框 Operations on selected triangles 中的 Extract cuboid 如圖 4.8，之後會出現 Primitive cuboid 的對話框並按下 OK 即會產生一個立方體如圖 4.9，頭部的簡化就完成了，再將簡化對話框關閉，此時會出現 Shape edit mode 對話框詢問是否應用這些變化嗎？(點擊否會保留提取的對象)，所以在這選擇 No 將新加入的物件保留。



圖 4.2: onshape 匯出對話框

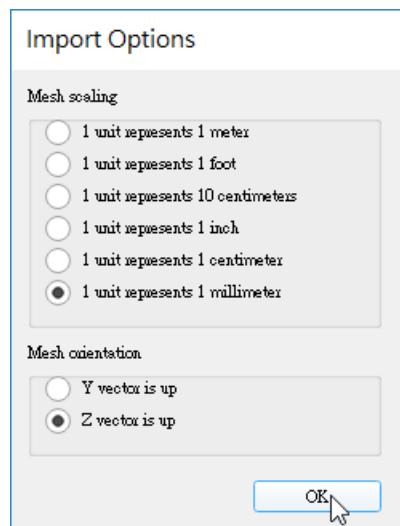


圖 4.3: 匯入.STL 檔後的對話框

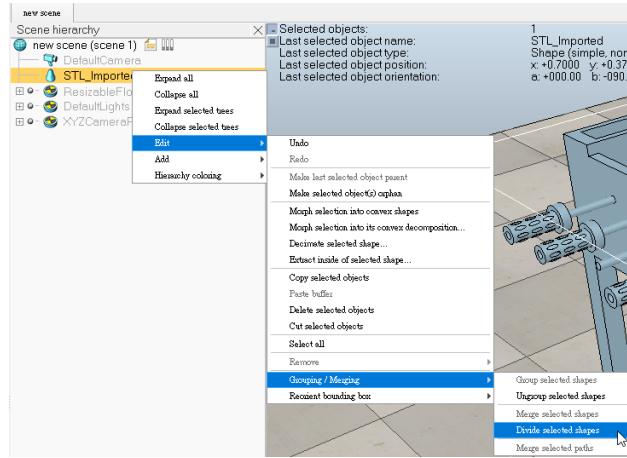


圖 4.4: 分離模型步驟點選

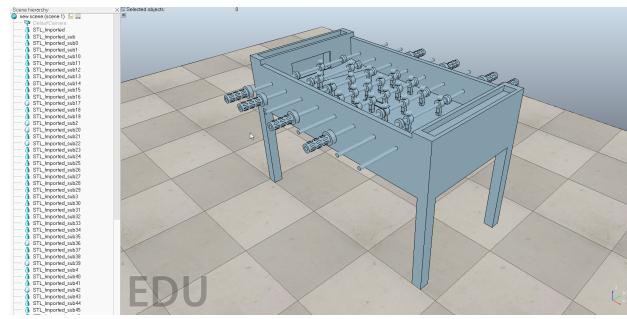


圖 4.5: 模型爆開後



圖 4.6: 頁面選擇器工具欄按鈕



圖 4.7: 形狀編輯模式工具欄按鈕

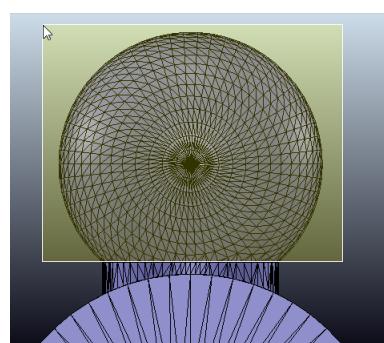


圖 4.8: 框選人物頭部

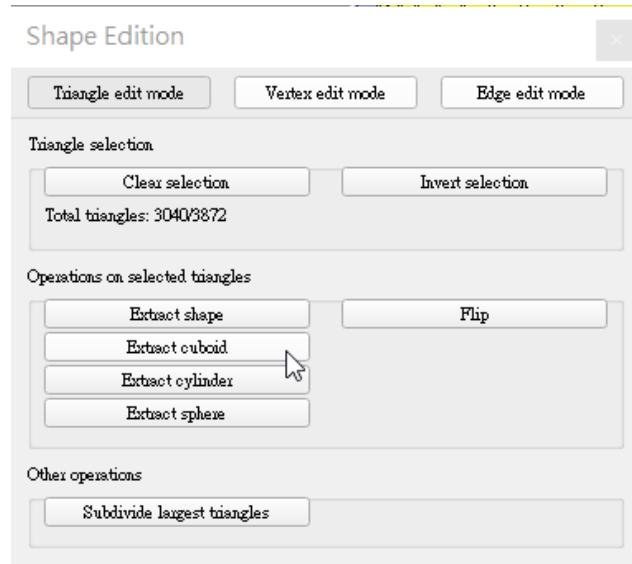


圖 4.9: 簡化的對話框 - 人物

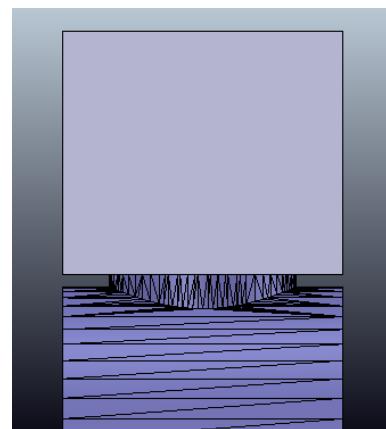


圖 4.10: 頭部簡化完成後

再來要說明人物身體的簡化，其簡化步驟和人物頭部簡化很相似，所以直接跳至身體簡化所需框選的部位如圖 4.10，之後再點選簡化的對話框 Operations on selected triangles 中的 Extract cuboid 如圖 4.8，身體簡化完的圖片如圖 4.11。

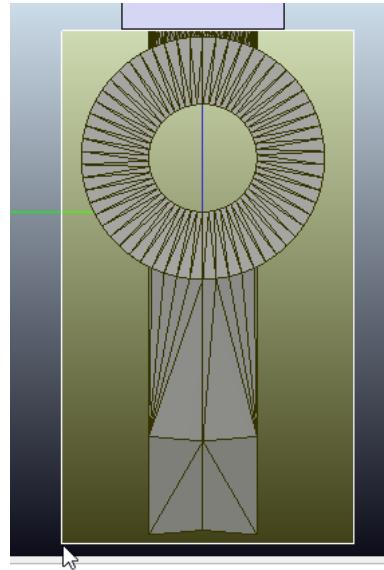


圖 4.11: 框選人物頭部以下的部位

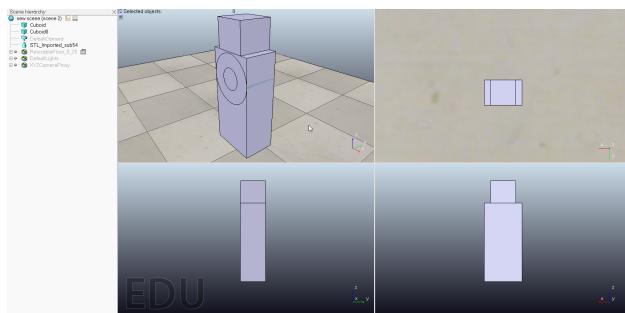


圖 4.12: 頭部簡化完成後

人物的簡化就到這，接著要將兩個新增的物件合併成一個全新的物件，先將兩個物件選取 (使用 [Edit -> Grouping/Merging -> Group selected shapes])，如圖 4.12，新物件的名稱為 people_dyn，之後在剪下貼回第一個場景，如圖 4.13。

其餘人物的簡化可以使用複製將剛剛簡化後的新物件複製在貼上，如圖 4.14，之後點選新貼上的物件使用 Ctrl 在選取其他尚未簡化的一個人物，如圖 4.15，點擊操作物件傳送工具欄按鈕如圖 4.16，會出現工具欄按鈕對話框，如圖 4.17，點選 Postion 中的 Apply to selection 使新貼上的物件能和為簡化的人物箱貼合，如圖 4.18，這樣就無需再簡化無限多次，完成後如圖 4.19。

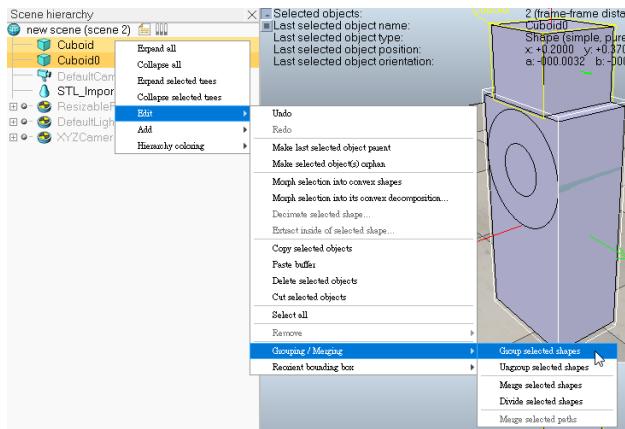


圖 4.13: 合併物件

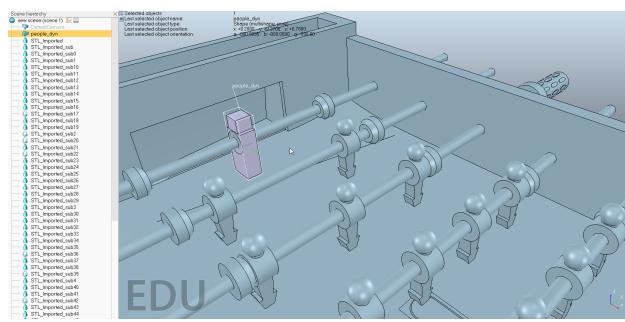


圖 4.14: 新物件 people_dyn 新加並貼回原位

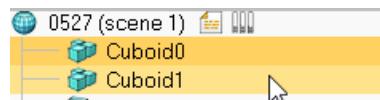


圖 4.15: 合併物件

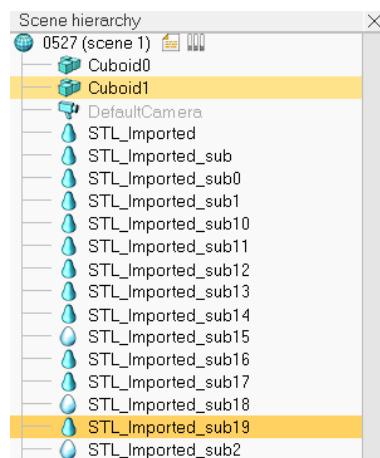


圖 4.16: 使用 ctrl 在點擊工具欄按鈕

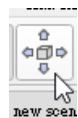


圖 4.17: 物件傳送工具欄按鈕

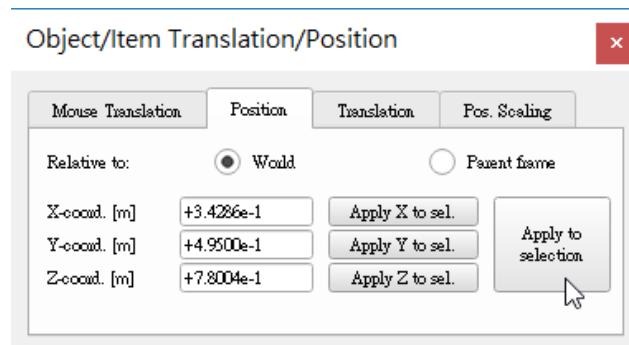


圖 4.18: 工具欄按鈕對話框

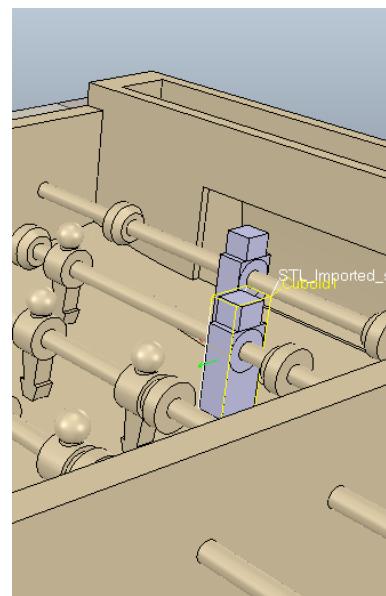


圖 4.19: 貼完後

4.2 軸、篩子簡化

在來先進行軸的簡化，複製軸到新的場景，選取軸並點選形狀編輯模式工具欄按鈕，如圖 4.6，將軸全部框選，如圖 4.20，再點選簡化的對話框 Operations on selected triangles 中的 Extract cylinder 如圖 4.21，之後會出現 Primitive cylinder 的對話框並按下 OK 即會產生一個圓柱如圖 4.22，將新圓柱物件命名為 Cylinder_dyn，並剪下貼回第一場景上即可。

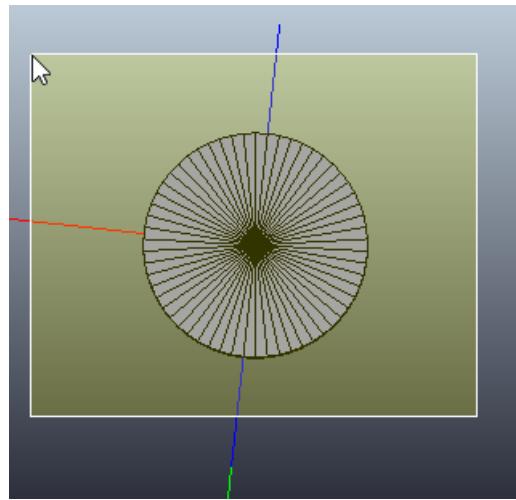


圖 4.20: 框選軸需簡化部分

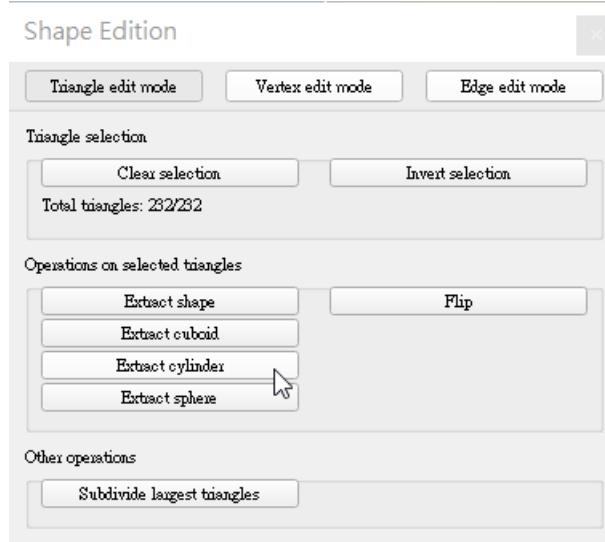


圖 4.21: 簡化對話框 - 軸

接下來要進行篩子的簡化，複製篩子至新的場景，之後的簡化也和軸非常相似，選取篩子並點選形狀編輯模式工具欄按鈕，將篩子全部框選進行簡化，如圖 4.23，

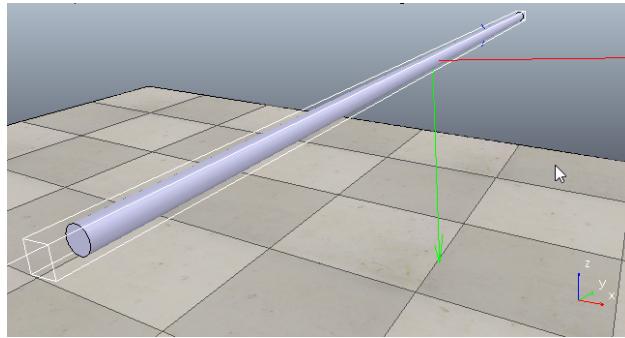


圖 4.22: 軸簡化後產生的新圓柱

再點選簡化的對話框 Operations on selected triangles 中的 Extract cylinder，之後會出現 Primitive cylinder 的對話框並按下 OK 即會產生一個圓柱如圖 4.24，將新圓柱物件命名為 s1_dyn，並剪下貼回第一場景上即可。

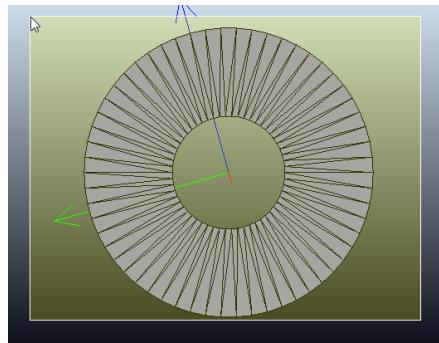


圖 4.23: 框選篩子需簡化部分

其餘的篩子及軸的簡化，可以參考人物簡化的最後一段。

4.3 足球桌簡化

影片簡易版

Toggle shape edit mode(左六) 僅對於“一個”物體作簡化形成多個三角形

先開啟另一個 New scene，將球桌複製過來做簡化。

球桌長板子有兩種方式，按 Toggle shape edit mode 後，

1. 為選取所想要的平面(可不須全選)，選取四角後按 Extract cuboid 後(系統會將所選所有三角形自動計算出最大尺寸)。

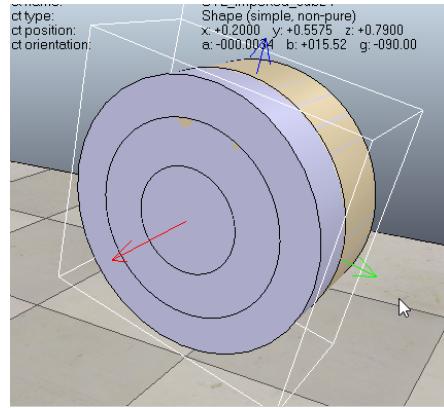


圖 4.24: 篩子簡化後產生的新物件

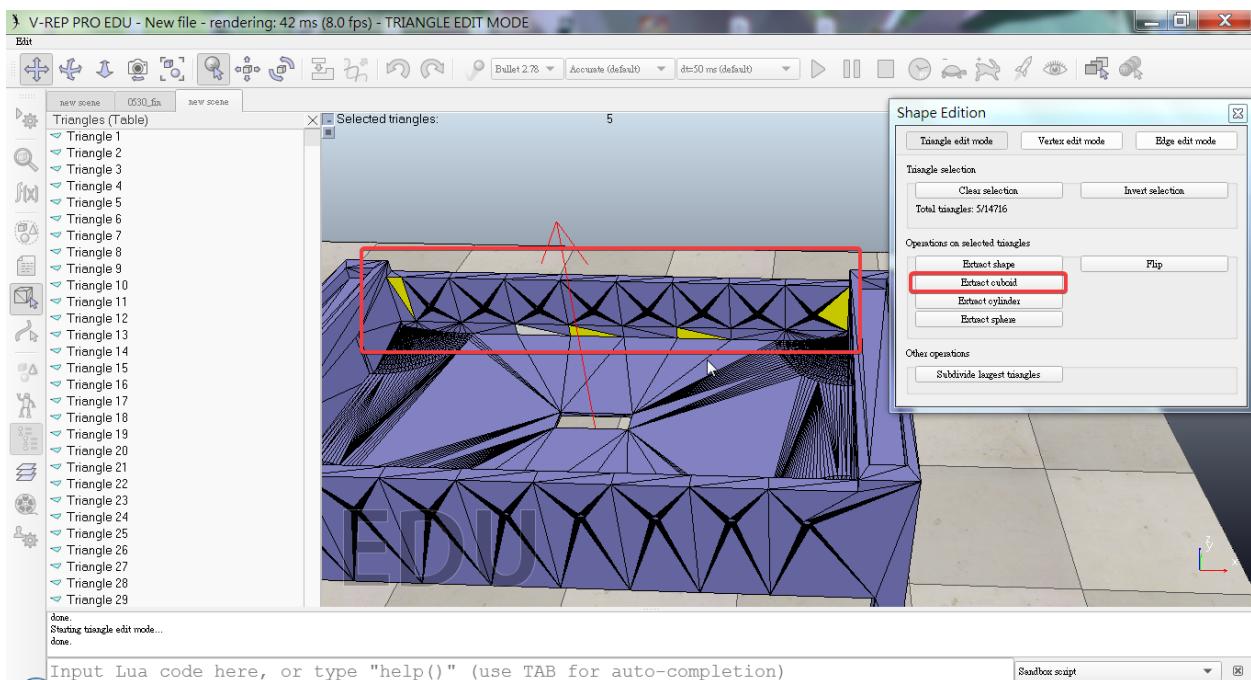


圖 4.25: 選取三角形後按 Extract cuboid

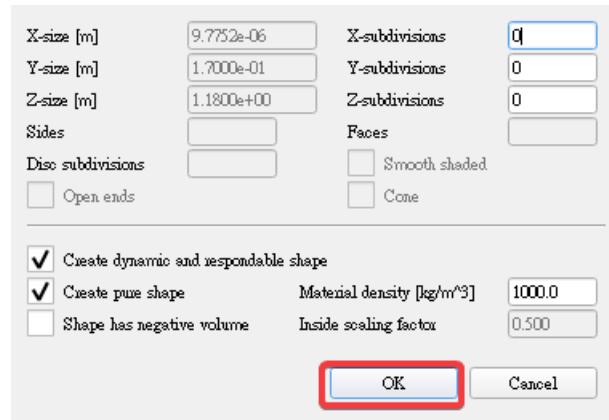


圖 4.26: OK

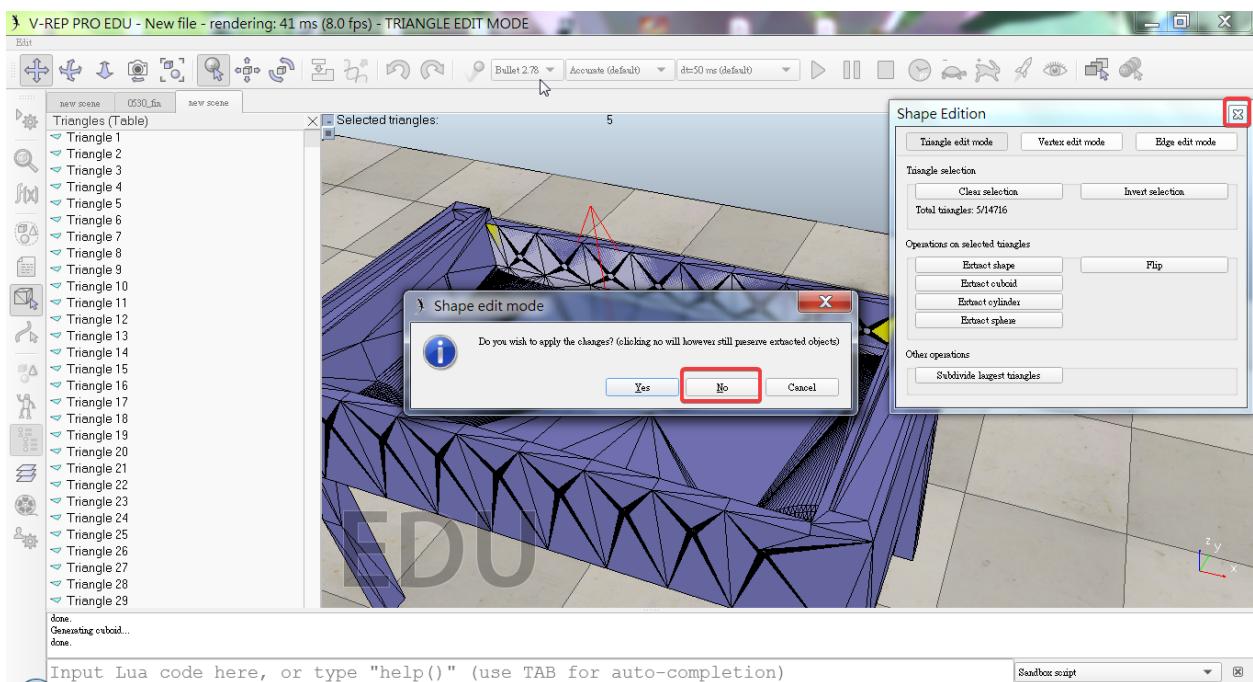


圖 4.27: 關閉 Shape Edition 後按 NO

Scene objects properties(左二)>Shape>View/modify geometry>Keep proportions 取消勾選，在 Bounding box size 長厚度後，在 Object/item shift(上七) 座移動，與球桌對齊。

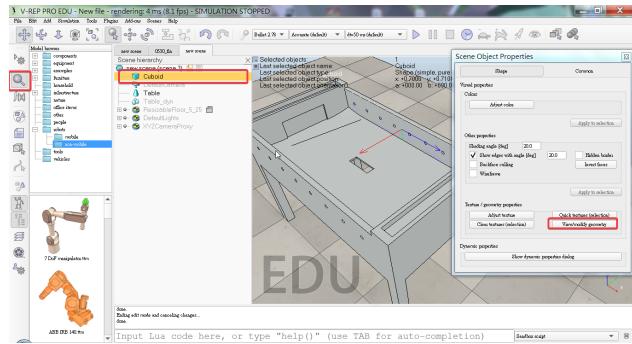


圖 4.28: View/modify geometry

2. 為選取所想要的平面(可不須全選)，選取四角後“再選取平面上方三角形(厚度)”Extract cuboid 後(系統會將所選所有三角形自動計算出最大尺寸)，就完成了。

選取完後一樣按 Extract cuboid>OK> 關閉 Shape Edition>NO

其他地方的板子操作也是如此，之後會發現板子相互干涉，但對模擬或是簡化不會有影響。

**

Extract shape 為類似像皮膚一樣的一層

Extract cuboid 為長平板

Extract cylinder 為長圓柱

Extract sphere 為長球

4.4 模擬步驟

將所有物件都簡化好後，開始進入模擬的步驟，為了在(場景層次結構)[<http://www.coppeliarobotics.com/helpFiles/en/userInterface.htm>]中的物件更好找到，將所有的物件都命名能更快的找的所需要的物件。首先先加入平移軸Prismatic joints (使用 [Add->Joint->Prismatic]) 及旋轉軸Revolute joints (使用 [Add->Joint-]

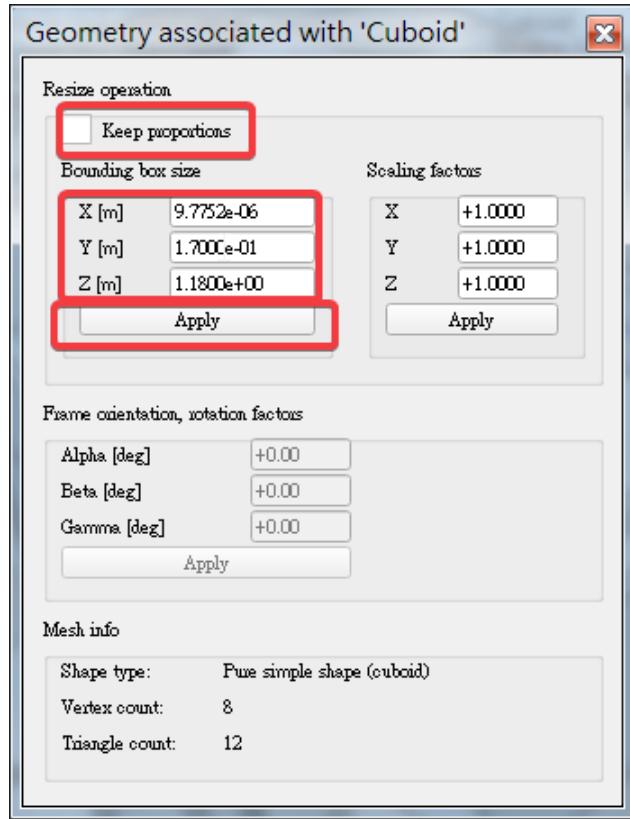


圖 4.29: 取消勾選並調整板厚

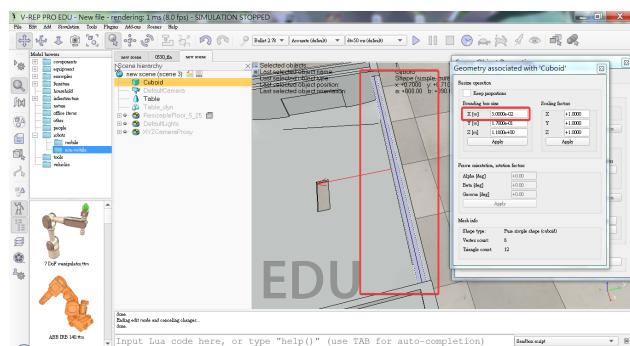


圖 4.30: X 方向調整

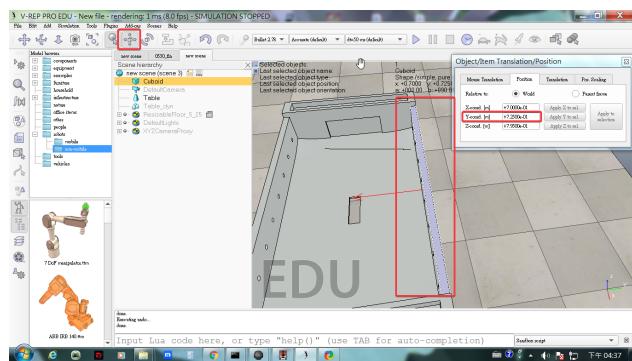


圖 4.31: 調整位置

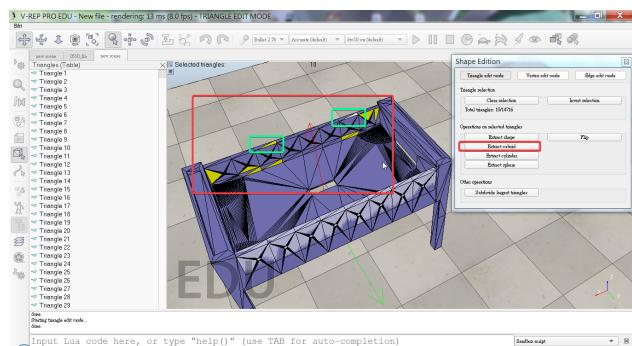


圖 4.32: 多選取板厚的三角形

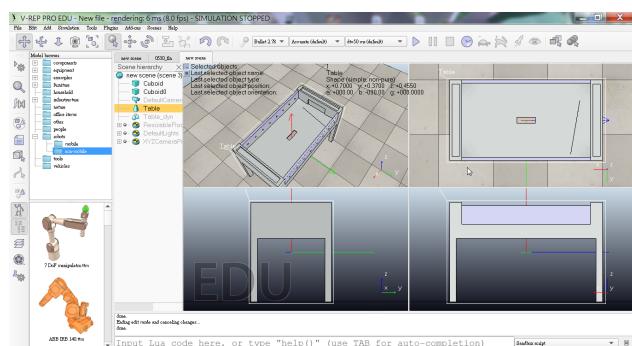


圖 4.33: 兩種方式厚度比較

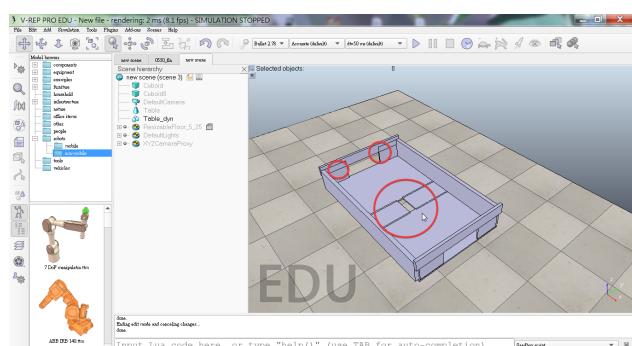


圖 4.34: 簡化後的板子相互干涉

>Revolute]), 並分別命名為 Prismatic_joint_1 及 Revolute_joint_1，之後將所有我需要的物件都拉到 Table_dyn 下，如圖 4.25 及圖 4.26。

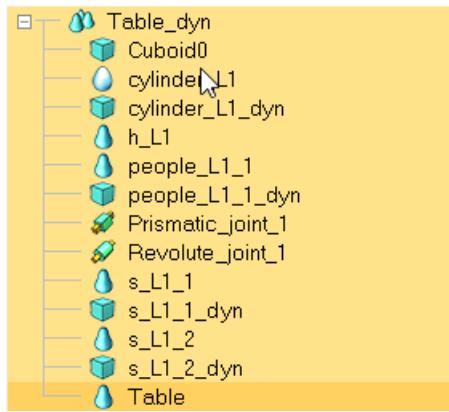


圖 4.35: 將所需物件拉入 Table_dyn 下-1

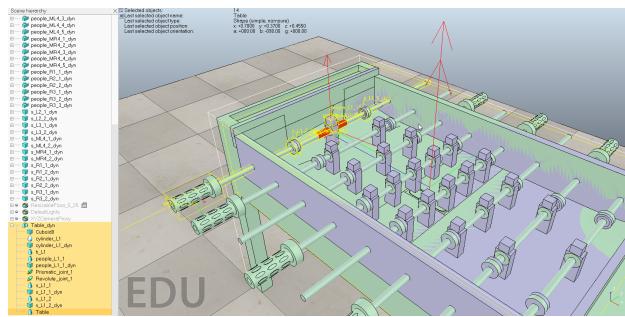


圖 4.36: 將所需物件拉入 Table_dyn 下-2

選取 Prismatic_joint_1 (平移軸) 及 cylinder_L1_dyn (軸) 並點選物件傳送工具欄按鈕如圖 4.16，會出現 Object/Item Translation/Position 的對話框，切換至 Postion 按下按鈕 Apply to selection，平移軸及軸會相貼合，如圖 4.27，再點選物件旋轉工具欄按鈕圖 4.28，會出現 Object/Item Rotation/Orientation 的對話框，切換至 Orientation 按下按鈕 Apply to selection，平移軸及軸方向會相同，如圖 4.29，之後的旋轉軸也是一樣的方法，圖 4.30 可以看出平移軸、旋轉軸及軸位置及方向都相同。

再將拉入 Table-dyn 下的物件依照圖 4.31 去排序，在選取物件 h-L1、people-L1-1、s-L1-1 及 s-L1-2 (使用 [Edit->Grouping/Merging->Group selected shapes]) 將物件合併，命名為 h-L1，另一部分的 people-L1-1-dyn、s-L1-1-dyn 及 s-L1-2-dyn 也一樣使用相同方法進行合併，命名為 s-L1-2_dyn，完成後如圖 4.32。

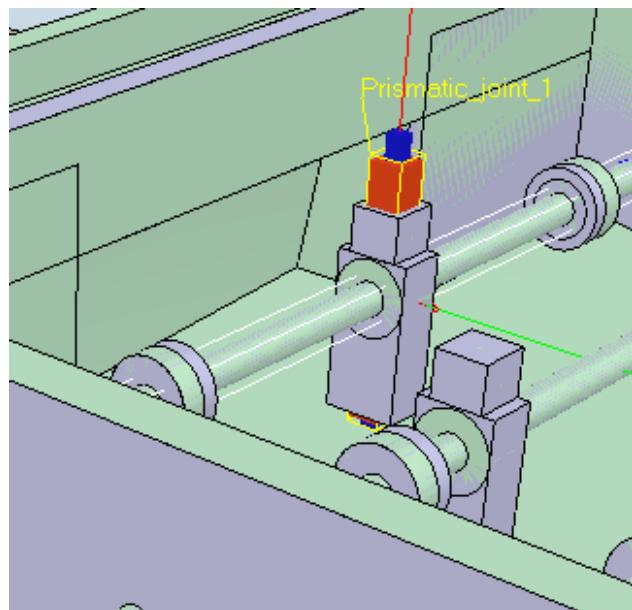


圖 4.37: 平移軸和軸相貼合



圖 4.38: 物件旋轉工具欄按鈕

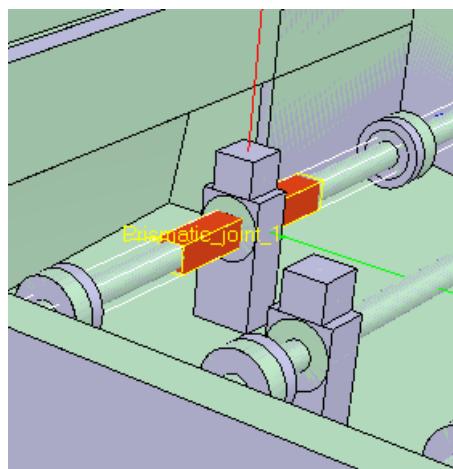


圖 4.39: 平移軸和軸方向相同

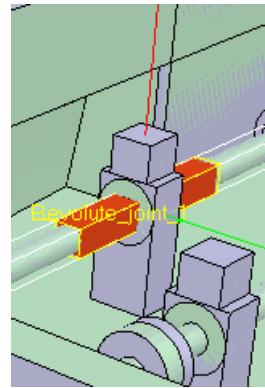


圖 4.40: 平移軸、旋轉軸及軸位置及方向皆一致

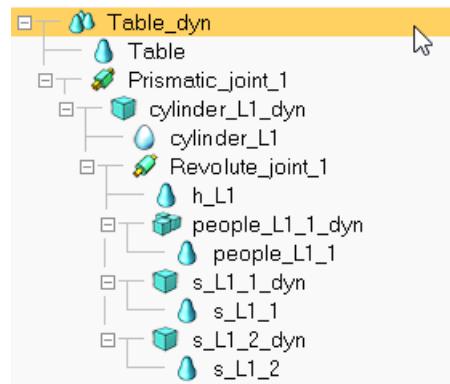


圖 4.41: 樹狀圖排列

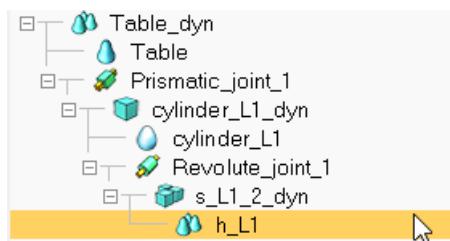


圖 4.42: 合併並排列好

之後點擊兩次 Table-dyn 前的圖案，會產生 Scene Object Properties 點選 common，將 Visibility 下的 Camera visibility layers 中的勾勾關掉，如圖 4.33，將 s-L1-2-dyn 及 cylinder-L1-dyn 也使用相同方式關閉圖層。

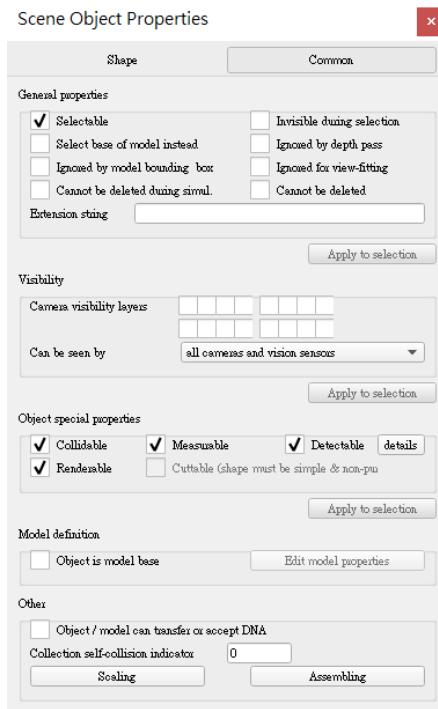


圖 4.43: Scene Object Properties 對話框

點擊兩次 Table-dyn 前的圖案，並點及對話框中的 Show dynamic properties dialog，會出現另一個 Rigid Body Dynamic Properties 對話框，將 Body is respondable 打勾，再將 Local respondable mask 從第 3 個勾開始不打勾到第 8 個，如圖 4.34；再來點擊兩次 cylinder-L1-dyn 前的圖案，依照圖 4.35 去進行勾選，其中打開 Body is dynamic 能使物體產生動態；再點擊兩次 s-L1-2-dyn 前的圖案，依照圖 4.36 去進行勾選。

接下來要讓軸能轉動，所以點擊兩次 Prismatic-joint-1 前的圖案，在按下 Show dynamic properties dialog，會產生 Joint Dynamic Properties 將 Motor properties 下的 Motor enabled 打勾，在 Target velocity 打入 0.001，如圖 4.37 所展示；再來點擊兩次 Revolute-joint-1 前的圖案，在按下 Show dynamic properties dialog，會產生 Joint Dynamic Properties 將 Motor properties 下的 Motor enabled 打勾，在 Target velocity 打入 1，如圖 4.38，最後看圖 4.39 可以看到模擬。

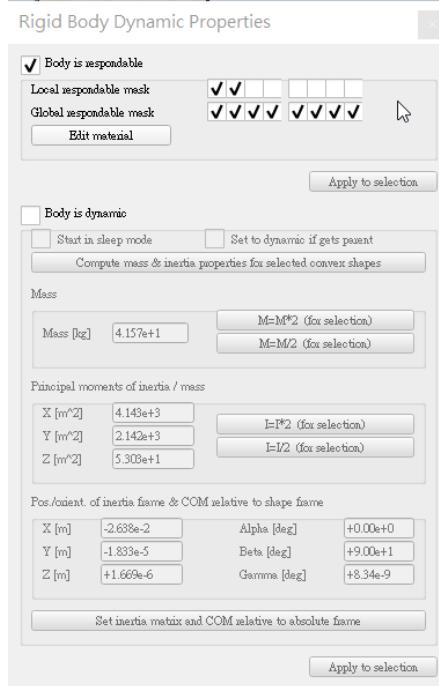


圖 4.44: 對話框勾選 - Table_dym

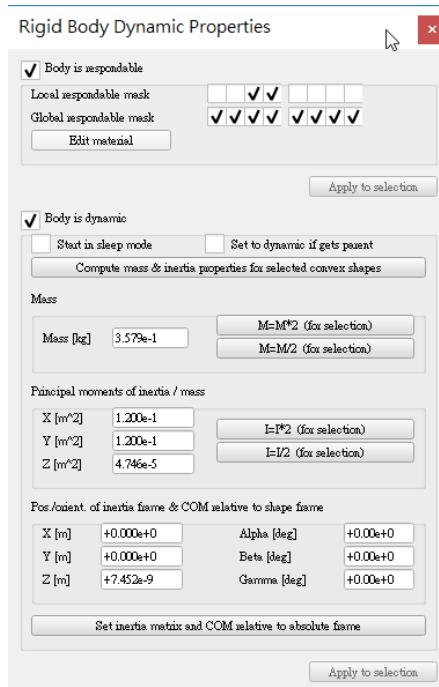


圖 4.45: 對話框勾選 - cylinder-L1-dyn

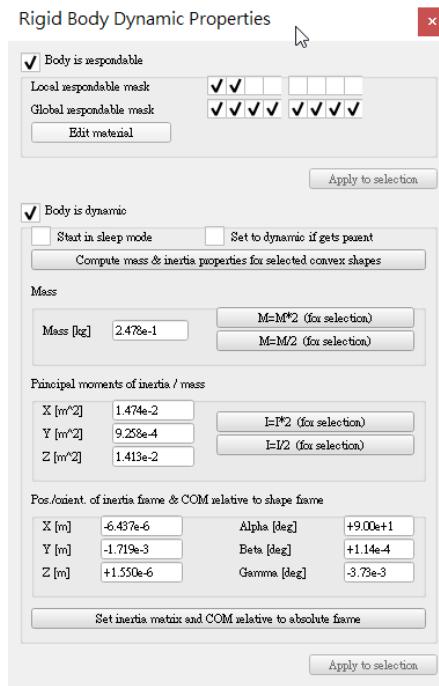


圖 4.46: 對話框勾選 - s-L1-2-dyn

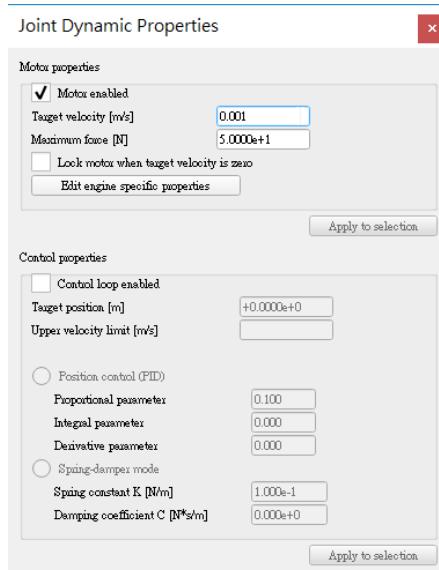


圖 4.47: 對話框勾選 - Prismatic-joint-1

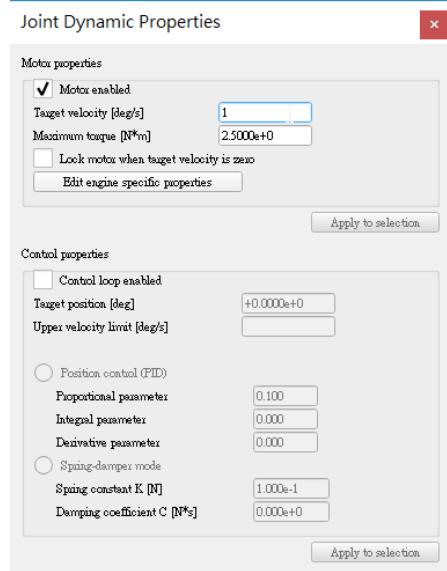


圖 4.48: 對話框勾選 - Revolute-joint-1

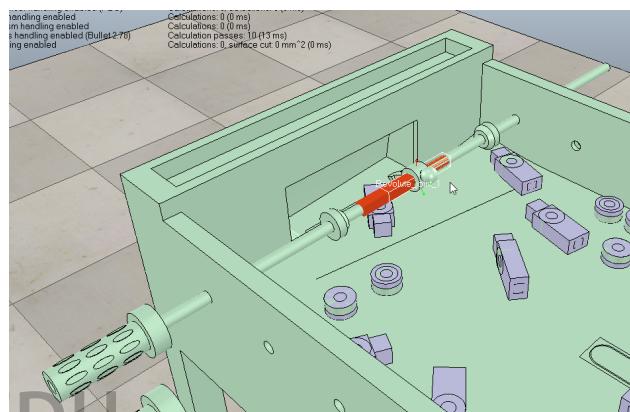


圖 4.49: 模擬圖

第五章 系統功能展示

5.1 雙人鍵盤控制對打

程式利用 python 的 keyboard module 來配和按鍵控制

所使用的是 keyboard.is_pressed('熱鍵'), 如果按下熱鍵, 則返回 True

並配合 Remote API functions (Python) 執行程式

```
while True:  
    try:  
        if keyboard.is_pressed('c'):  
            vrep.simxSetJointTargetVelocity(clientID,BRev_handle,B_KickBallVel,vrep.simx_opmode_oneshot_wait)  
        elif keyboard.is_pressed('v'):  
            vrep.simxSetJointTargetVelocity(clientID,BRev_handle,R_KickBallVel,vrep.simx_opmode_oneshot_wait)  
        elif keyboard.is_pressed('z'):  
            vrep.simxSetJointTargetVelocity(clientID,BMo_handle,0.2,vrep.simx_opmode_oneshot_wait)  
        elif keyboard.is_pressed('x'):  
            vrep.simxSetJointTargetVelocity(clientID,BMo_handle,-0.2,vrep.simx_opmode_oneshot_wait)  
        else:  
            vrep.simxSetJointTargetVelocity(clientID,BMo_handle,0,vrep.simx_opmode_oneshot_wait)  
    except:  
        break  
    try:  
        if keyboard.is_pressed('o'):  
            vrep.simxSetJointTargetVelocity(clientID,RRev_handle,R_KickBallVel,vrep.simx_opmode_oneshot_wait)  
        elif keyboard.is_pressed('p'):  
            vrep.simxSetJointTargetVelocity(clientID,RRev_handle,B_KickBallVel,vrep.simx_opmode_oneshot_wait)  
        if keyboard.is_pressed('u'):  
            vrep.simxSetJointTargetVelocity(clientID,RMo_handle,0.1,vrep.simx_opmode_oneshot_wait)  
        elif keyboard.is_pressed('i'):  
            vrep.simxSetJointTargetVelocity(clientID,RMo_handle,-0.1,vrep.simx_opmode_oneshot_wait)  
        else:  
            vrep.simxSetJointTargetVelocity(clientID,RMo_handle,0,vrep.simx_opmode_oneshot_wait)  
    except:  
        break  
    MMMB = Bv*2  
    MMMR = Rv*2
```

圖 5.1: 玩家對打

5.2 雙電腦對打

一開始發現問題如果球再球員擊出後, 跑到桿子後方, 會出現 Bug 球員將無法收回

經過修改後再擊出球後能往旁邊移動收回桿子, 進行下段程式。

如果球進入擊球區桿子在收回的情況下，球與球員進入 x 軸-0.01~0.01 範圍內
則球員會將球擊出

未進入 x 軸-0.01~0.01 範圍內時則桿子保持收回

未進入擊球範圍時桿子繼續保持收回

```
if position_S[0] >= 1.185 and RRev_deg >= 25: #球進入擊球區 and 桿子擊出 紅色桿子
    if position_RR[1]>=0.5: #桿子偏右邊
        speed(RMo_handle, -0.08)
        sleep(0.2)
        speed(RRev_handle, B_KickBallVel)
        sleep(0.2)
    elif position_RR[1] < 0.5: #桿子偏左邊
        speed(RMo_handle, 0.08)
        sleep(0.2)
        speed(RRev_handle, B_KickBallVel)
        sleep(0.2)
    elif position_S[0] >= 1.185 and RRev_deg <=0: #球進入擊球區 and 桿子收回
        if Rv>= -0.01 and Rv <0.01: #進入左右擊球範圍
            speed(RRev_handle, R_KickBallVel)
        else:
            speed(RRev_handle, B_KickBallVel)
    elif position_S[0] < 1.185: #球未進入擊球區
        speed(RRev_handle,B_KickBallVel)

if position_S[0] <= 0 and BRev_deg <= -25: #球進入擊球區 and 桿子擊出 藍色桿子
    if position_BR[1]<=0.5: #桿子偏右邊
        speed(BMo_handle, 0.08)
        sleep(0.2)
        speed(BRev_handle, R_KickBallVel)
        sleep(0.2)
    elif position_BR[1] > 0.5: #桿子偏左邊
        speed(BMo_handle, -0.08)
        sleep(0.2)
        speed(BRev_handle, R_KickBallVel)
        sleep(0.2)
    elif position_S[0] <= 0 and BRev_deg >=0:
        if Bv>= -0.01 and Bv <0.01:
            speed(BRev_handle, B_KickBallVel)
        else:
            speed(BRev_handle ,R_KickBallVel)
    elif position_S[0] > 0:
        speed(BRev_handle,R_KickBallVel)
```

修正舊版在桿子擊出後無法回擊錯誤
判定桿子絕對位置在左右半場後
再決定向左向右收起桿子

當球與球員y軸距離
進入-0.01~0.01內擊出
未進入則保持桿子收回

5.3 單人鍵盤控制與電腦對打

由電腦對打和玩家對打程式合併加以修改

完整程式碼如圖 5.3

```
while True:
    if position_S[0] >= 1.185 and RRev_deg >= 25: #球進入擊球區 and 棍子擊出
        if position_RR[1]>=0.5: #棍子偏右邊
            speed(RMo_handle, -0.08)
            sleep(0.2)
            speed(RRev_handle, B_KickBallVel)
            sleep(0.2)

        elif position_RR[1] < 0.5: #棍子偏左邊
            speed(RMo_handle, 0.08)
            sleep(0.2)
            speed(RRev_handle, B_KickBallVel)
            sleep(0.2)

    elif position_S[0] >= 1.185 and RRev_deg <=0: #球進入擊球區 and 棍子收回
        if Rv>= -0.01 and Rv <0.01: #進入左右擊球範圍
            speed(RRev_handle, R_KickBallVel)

        else:
            speed(RRev_handle, B_KickBallVel)

    elif position_S[0] < 1.185: #球未進入擊球區
        speed(RRev_handle,B_KickBallVel)

    try:
        if keyboard.is_pressed('o'):
            vrep.simxSetJointTargetVelocity(clientID,RRev_handle,R_KickBallVel,vrep.simx_opmode_oneshot_wait)
        else:
            vrep.simxSetJointTargetVelocity(clientID,RRev_handle,B_KickBallVel,vrep.simx_opmode_oneshot_wait)
        if keyboard.is_pressed('u'):
            vrep.simxSetJointTargetVelocity(clientID,RMo_handle,0.1,vrep.simx_opmode_oneshot_wait)
        elif keyboard.is_pressed('i'):
            vrep.simxSetJointTargetVelocity(clientID,RMo_handle,-0.1,vrep.simx_opmode_oneshot_wait)
        else:
            vrep.simxSetJointTargetVelocity(clientID,RMo_handle,0,vrep.simx_opmode_oneshot_wait)
    except:
        break
```

圖 5.3: 玩家電腦

5.4 影像辨識

利用 vrep 的 vision sensor 來拍攝模擬畫面，並輸出至外部 python 程式中進行影像處理。

首先進行影像模糊化來降低雜訊

處理完的圖片再進行顏色處理把 RGB 轉為

處理完後運用 opencv 的 canny 來尋找物體邊界，並用找到的邊界計算形心位置。

關於 opencv-canny

這個函數有五大步驟分別是

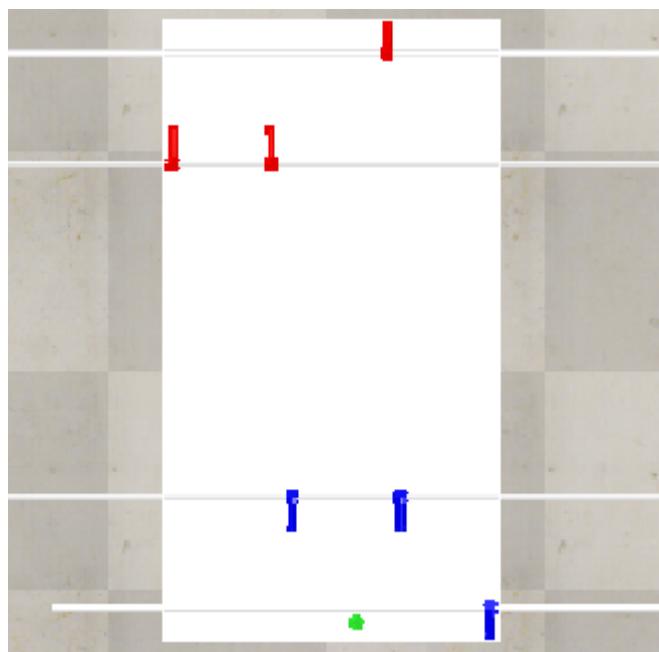


圖 5.4: 原圖



圖 5.5: 模糊處理後

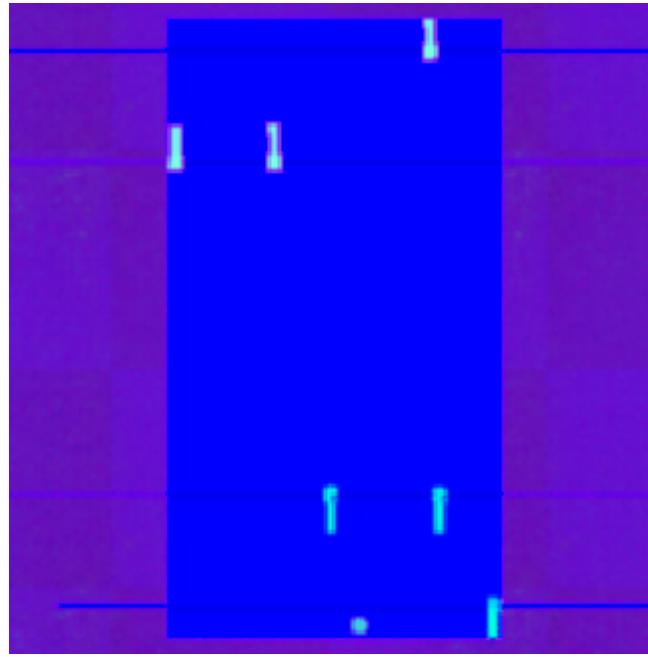


圖 5.6: hsv 處理後

5.5 影像辨識-canny

canny 函數有四大步驟分別為 1. 高斯濾波 (Gaussian filter)

矩陣中的值代表對應像素值的加權 (越往旁邊加權越小越中間越大)

而分數分母為加權值的總合

$$K = \frac{1}{159} \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

圖 5.7: 高斯濾波

2. 尋找影像的強度斜率

3. 刪除不是邊緣的像素

4. 從定義的數值決定邊線是刪除或留下 (上限和下限)(建議 2:1~3:1 之間)

a. 如果高於上限留下

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

圖 5.8: 尋找 G_x 與 G_y



圖 5.9: 找到強度斜率與方向

- b. 介於上限與下限之間時至有當它連接 a. 時才留下
- c. 低於下限時則刪除它

5.6 影像辨識電腦對打

利用上一節所得形心座標判斷各球員與球的相對運動，再利用判斷的結果控制球員該往哪裡移動，或是否該踢球（整體架構如下圖）。

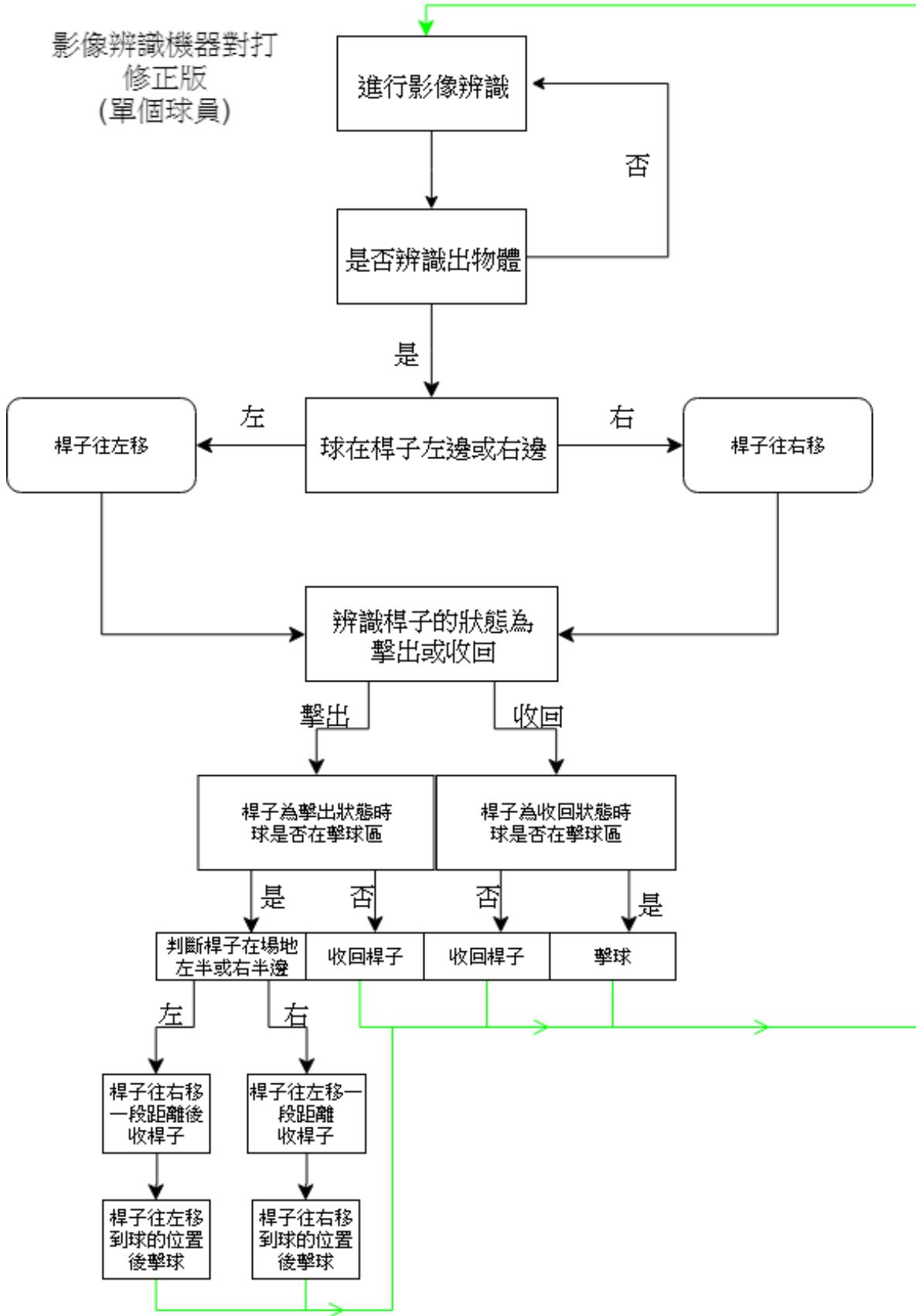


圖 5.10: 守門員程式架構
50

第六章 參考文獻

vrep-opencv:<https://github.com/nemilya/vrep-api-python-opencv>

opencv-canny:https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

高斯濾波:https://en.wikipedia.org/wiki/Gaussian_filter

影像的強度斜率:<https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm>