國立虎尾科技大學

機械設計工程系

電腦輔助設計實習 ag2 手足球

期末結報

Final Report

學生：

設計二甲陳鉅忠 40623130

設計二甲吳隆廷-40623115

設計二甲蕭家瀚-40623133

設計二甲許高惟-40623139

設計二甲郭益綸-40623142

設計二甲林暉恩-40623145

設計二甲劉奇 -40623146

設計四甲陳柏維-40423136

設計四乙周政叡-40423218

指導教授：嚴家銘

摘要

執行規劃

手足球系統模擬

設計與繪圖

零組件尺寸分析

參數設計與繪圖

細部設計與 BOM

V-rep 動態模擬

送球機構設計與模擬

系統功能展示

目錄

表目錄

圖目錄

第一章 前言

V-rep 對於一個機械程式設計者而言是不錯的選擇，可編譯可模擬，檔案又不會太大，可以依照個人需求更改各式各樣的設定，而這份 PDF 主要就是要介紹手足球於 V-rep 中的設定與編譯。

包含了

(1)零件尺寸分析

(2)設計與繪圖

(3)機構設定

(4)編譯Lua

(5)了解運動原理

第二章 執行規劃

執行規劃為每週安排任務，計畫通常趕不上變化，而變化通常趕不上長官的一句話:

W11 - 開會討論工作分配以及之後各週目標。

W12 - 利用 Onshape 設計並將各零件及場地繪製完成並導出、簡化導出圖檔。

W13 - 利用 Onshape 設計並將各零件及場地繪製完成並導出、利用 Lua 測試並完成手足球 1 對 1 回擊。

W14 - 於 Vrep 設定回球機構、利用 Lua 測試並完成手足球玩家對電腦。

W15 - 結合全部物件測試 & 修改

W16 - 整理並開始編輯個人網站 & 小組網站 &PDF& 影片。

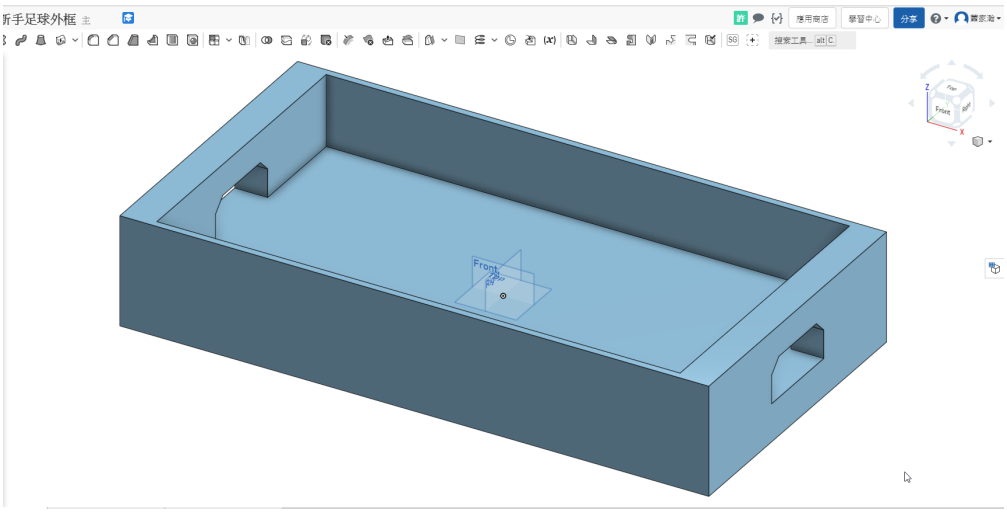
W17 - 在課堂上進行分組報告。

第三章 設計與繪圖

3.1 零組件尺寸分析

手足球場地

1. 球門及場地的大小，參考網路上標準場地的尺寸來做設計。但由於在模擬時發現球門前的球員動作距離有限, 球容易卡在死角, 最後決定將球門牆壁加厚



球場

2. 球員跟竿子尺寸，參考網路上標準的尺寸來做設計。但竿子的長度有做調整，方便模擬球員如圖 3.1 竿子如圖 3.2

3. 軌道最終版

由於我們的擊球系統過於強大，會讓球從舊版軌道直接飛出去，所以才把軌道上面封起來，變成最終版

軌道如圖 3.3

4. 擊球系統配合軌道使用如圖 3.4

5. 導球球門球進之後，將球導到擊球系統，出並沿著軌道回到場上

如圖 3.5

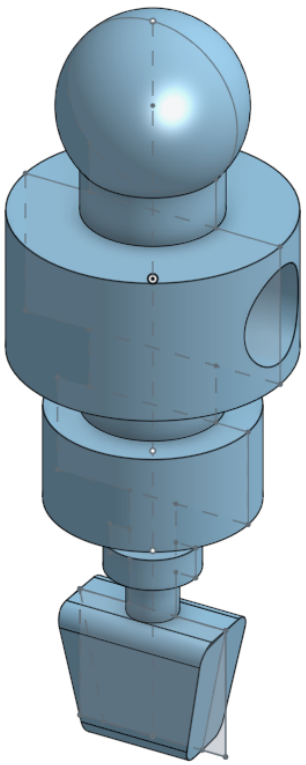


圖 3.1:

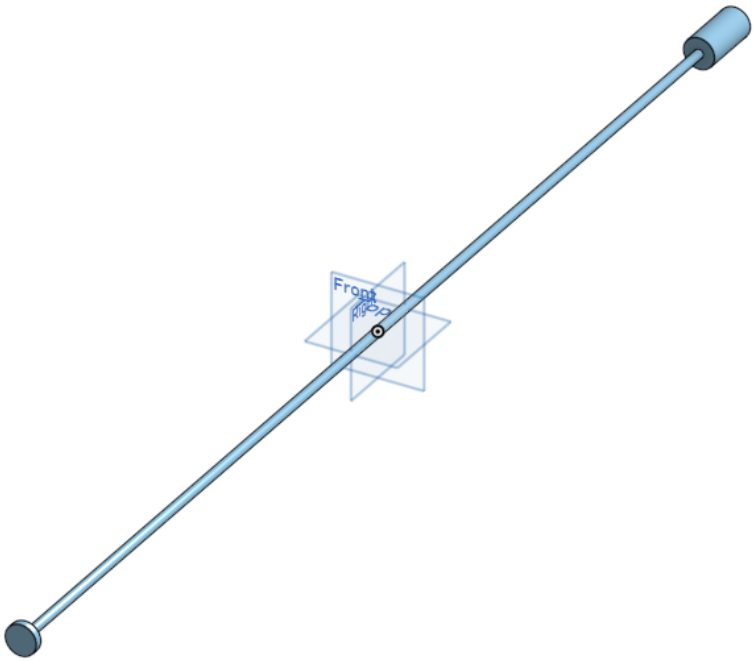


圖 3.2:

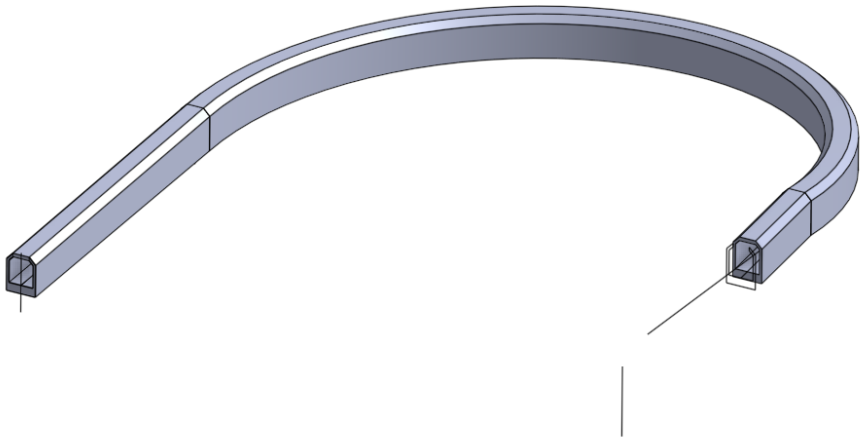


圖 3.3:

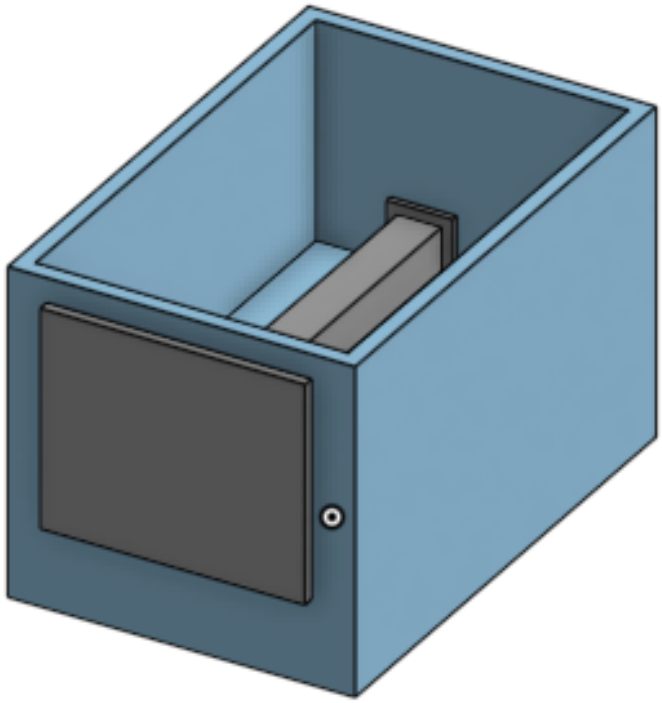


圖 3.4:

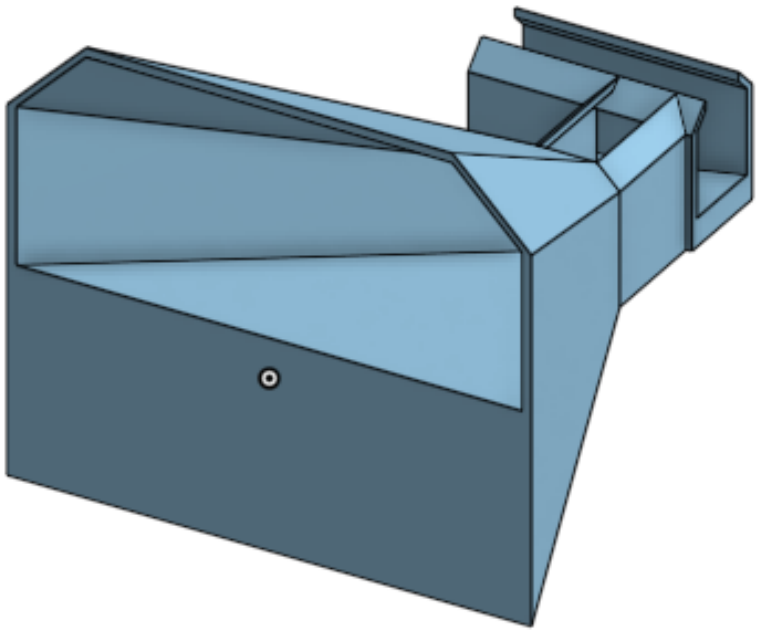


圖 3.5:

3.2 參數設計與繪圖

手足球系統的零組件參數設計與繪圖 (零組件初步設計繪圖)

Onshape 零組件連結

【機構設計】

〈初始設計〉

初始設計挑出了選多方案，如:

1. 桿件推送

2. 螺旋尺上推

3. 打擊軌道

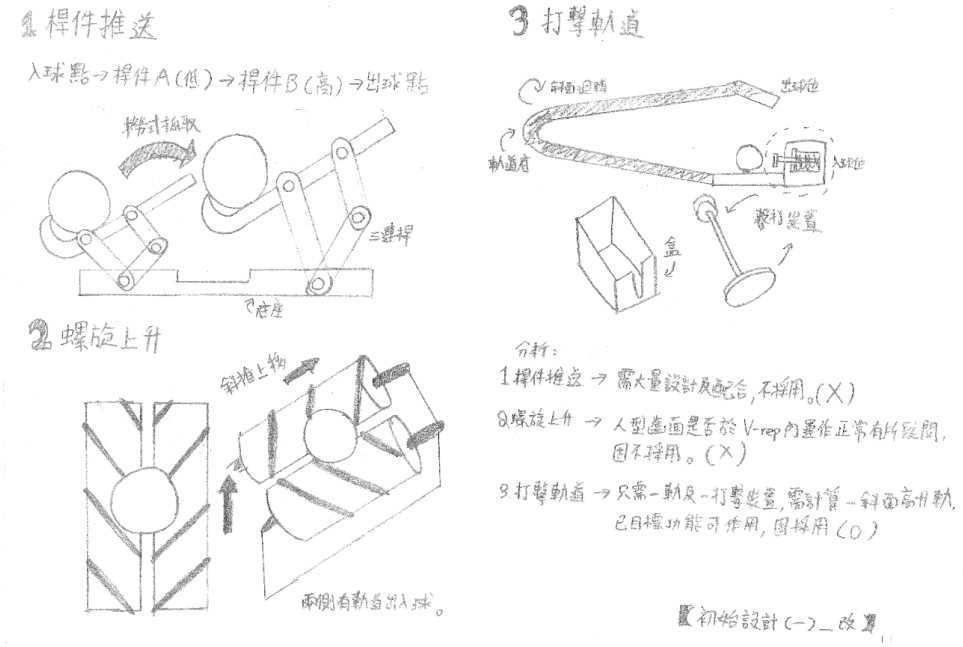


圖 3.6:

最後挑選打擊裝置搭配配斜面軌道的方式將球送出。

因只需一次做動就可完成目標。

〈軌道設計〉

最初設計是想將兩條軌道集合一束，用一次打擊就可，

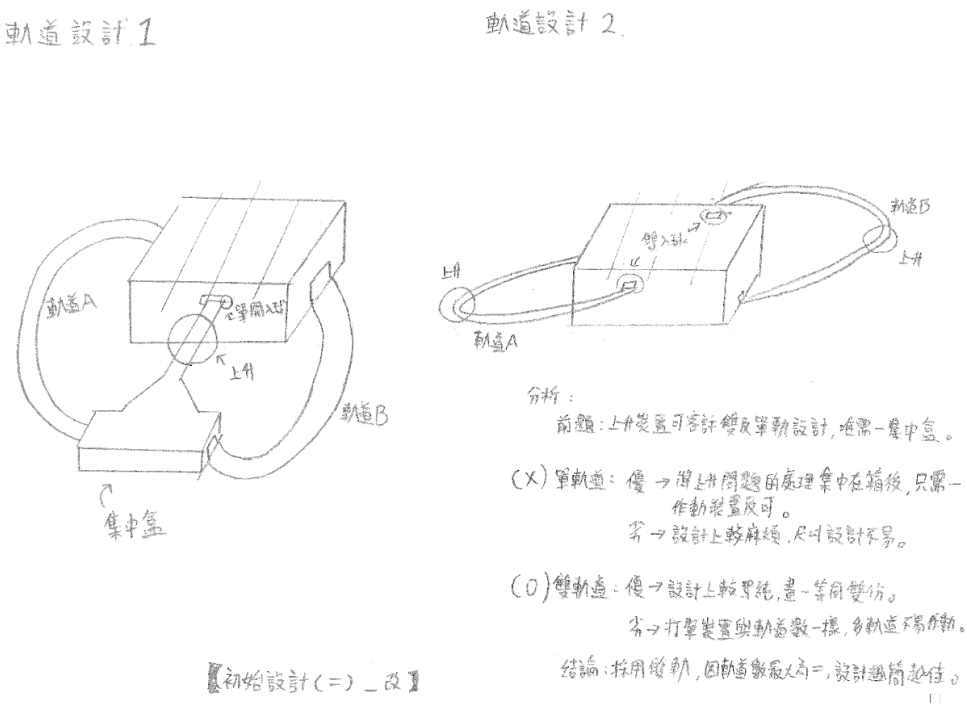


圖 3.7:

但後來發現會有收束誤差與繪製上的困難，後來還是改

為左右各開一條軌道，後來還是改為左右各開一條軌道

並各自擁有一打擊裝置。

最後採用複斜面旋轉軌道，將球送至最高點後，再使用斜面將球滾落。

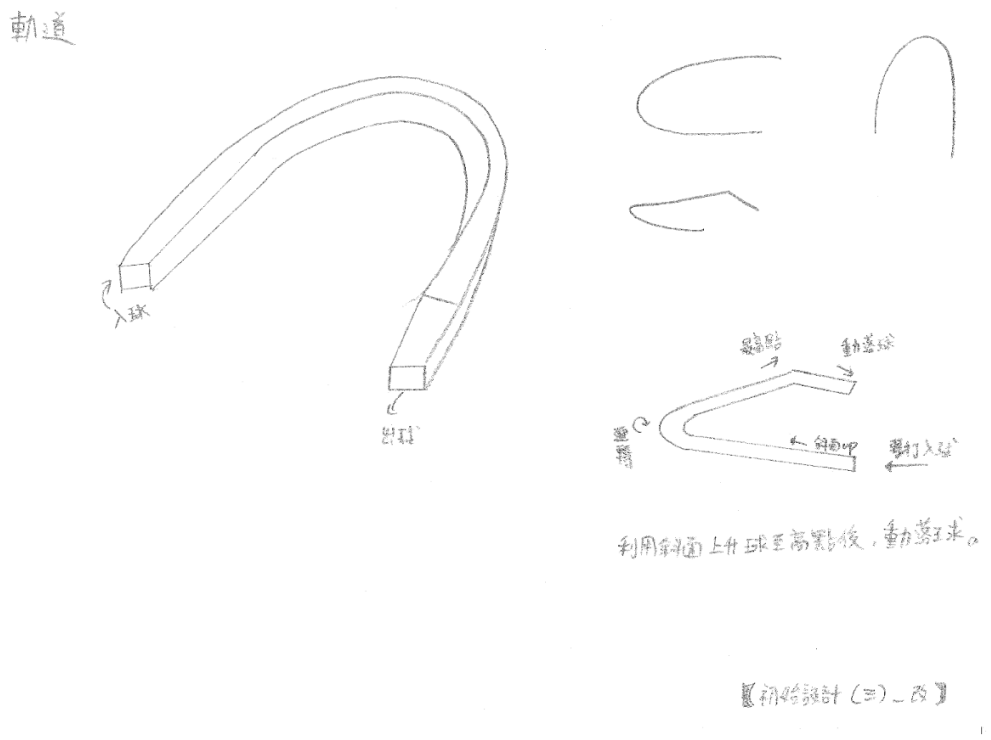
〈打擊機構設計〉

打擊機構非常直覺，就是使用一個帶緣圓棒將球打擊出去，

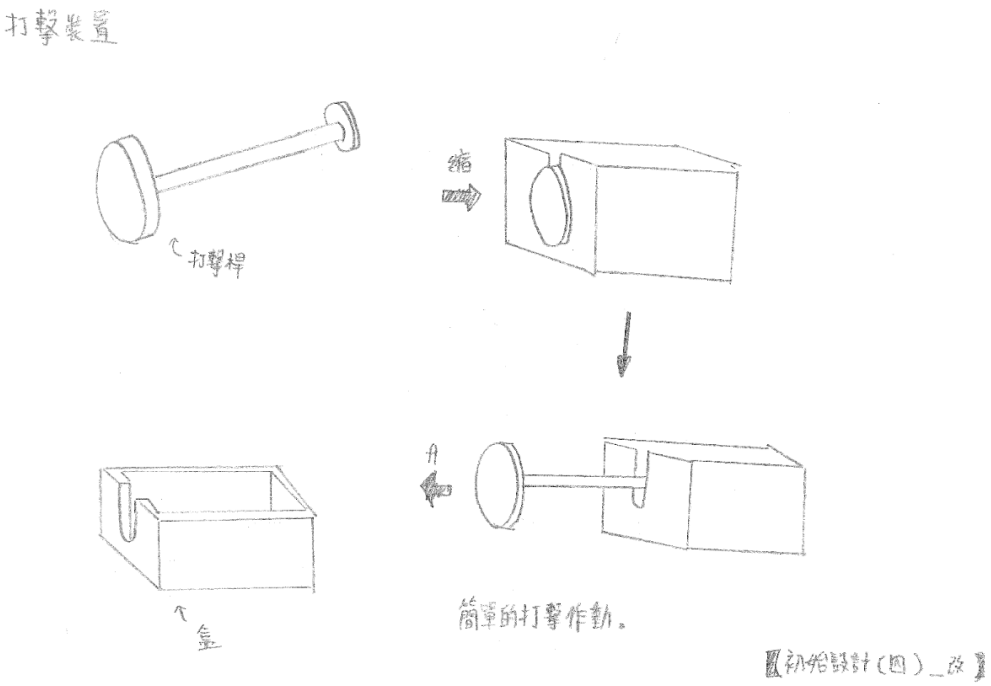
並外掛置打擊區旁，作為球之動力。

〈集球門〉

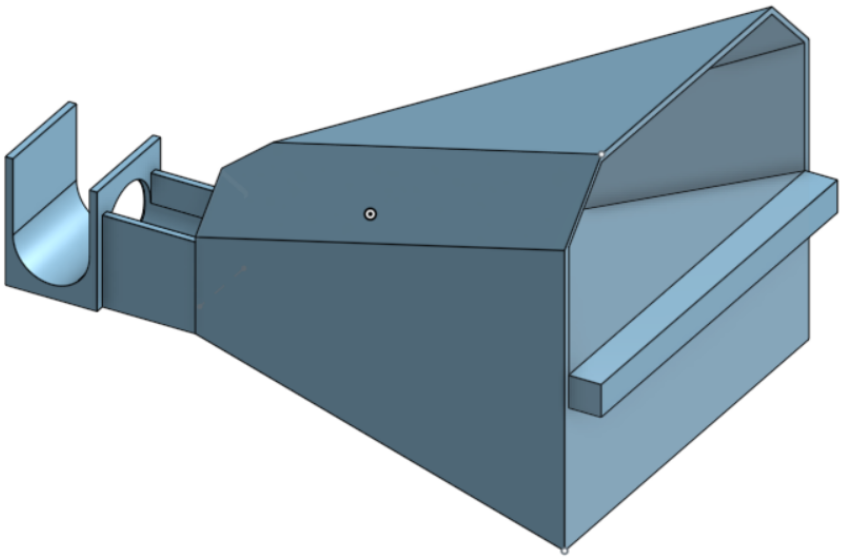
原先的設計，發現組裝後門框的高度沒有高於球門，怕會出現卡球的可能所以要重新畫，而假如球快速的撞擊檔板可能會造成球直接反彈並有可能回到場地內，就把球檯加高 3 英吋，在檔板的部分畫成密閉式以防球會跑出去


軌道
入球
出球
利用斜面上升球至高點後，重力落下來。
【初始設計（三）_改】

軌道


打擊裝置
打擊桿
縮
盒
簡單的打擊作動。
【初始設計（四）_改】

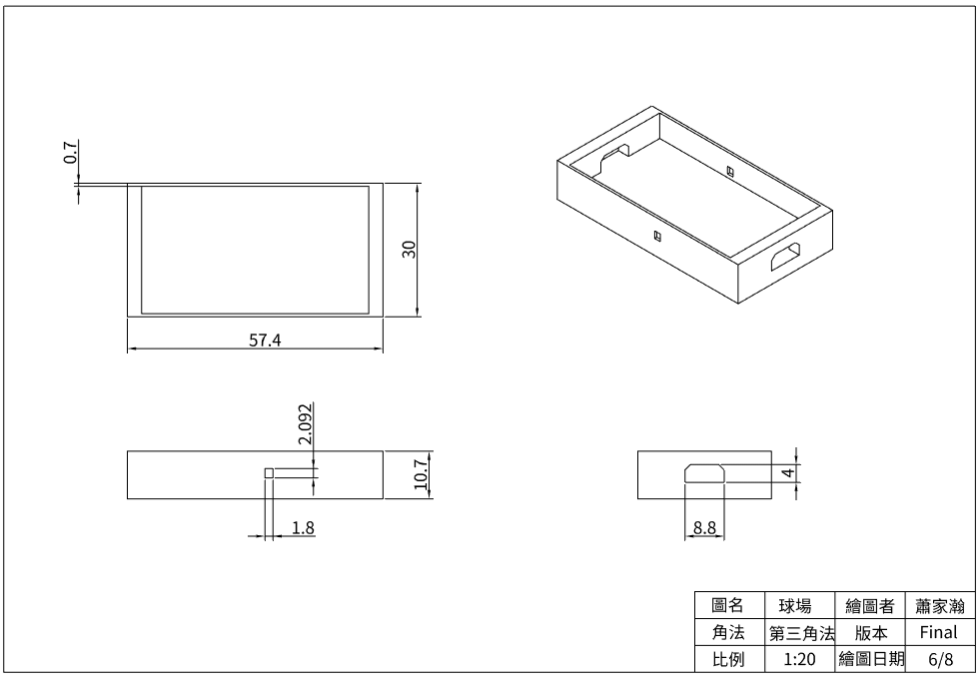
打擊機構設計



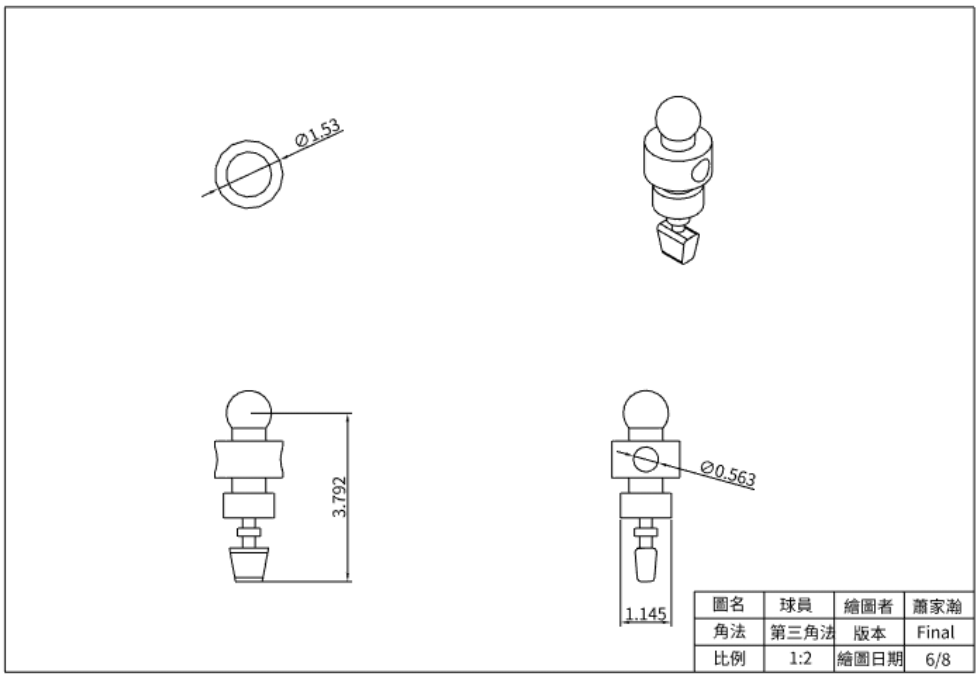
. .

3.3 細部設計與 BOM

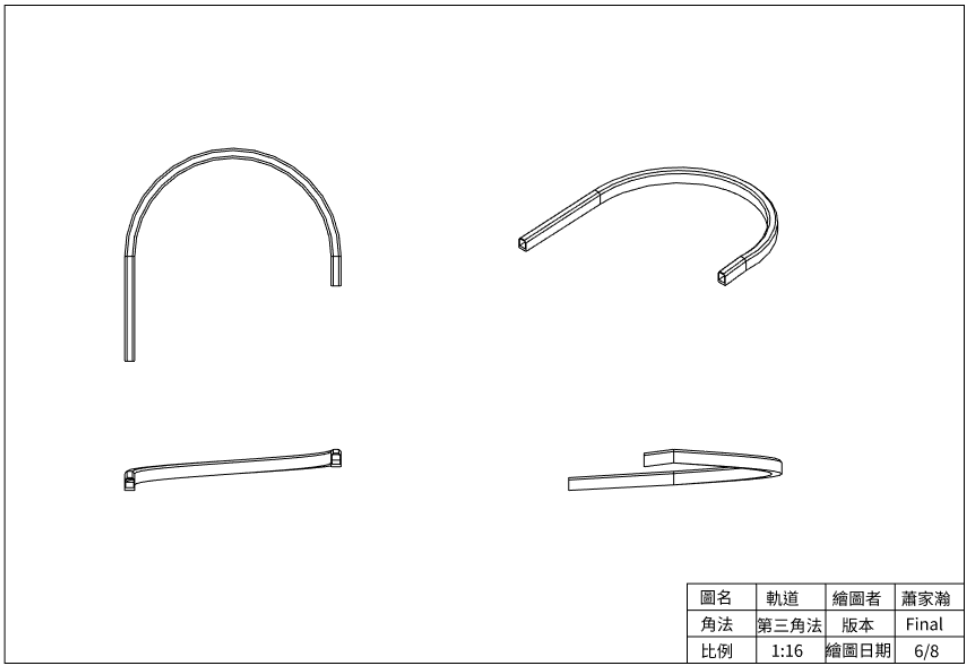
各零件圖 BOM



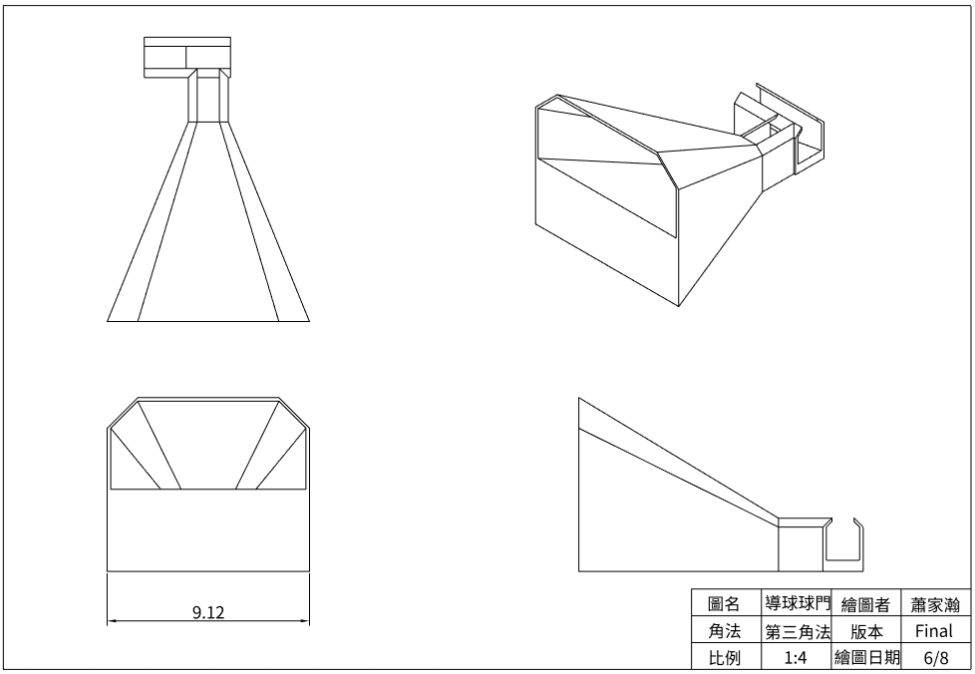
球場


Ø1.53
3.792
Ø0.563
1.145
圖名 球員 繪圖者 蕭家瀚
角法 第三角法 版本 Final
比例 1:2 繪圖日期 6/8

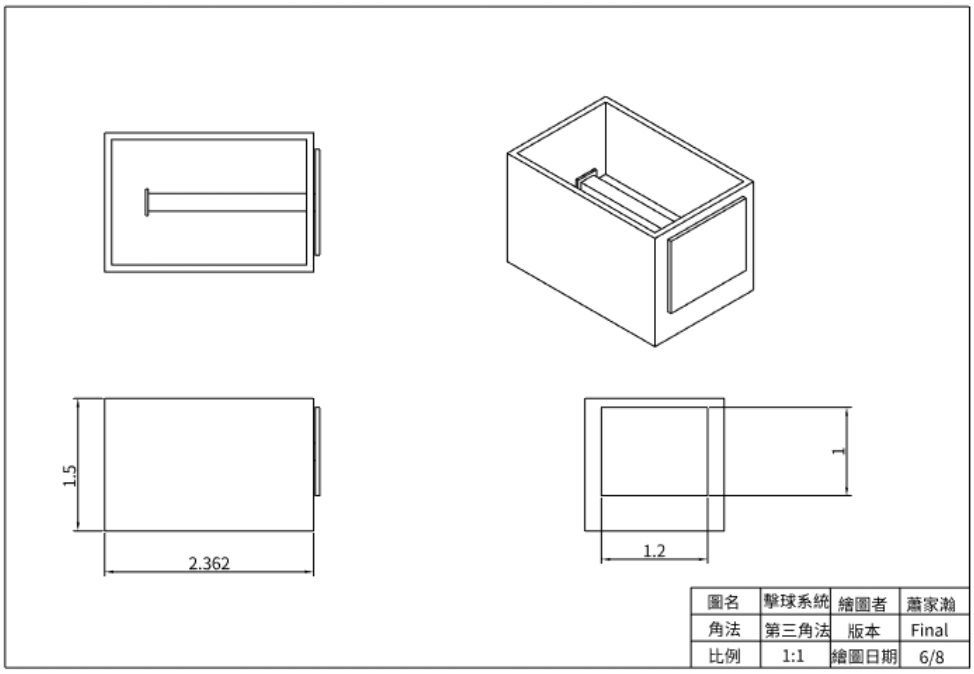
球員


圖名 軌道 繪圖者 蕭家瀚
角法 第三角法 版本 Final
比例 1:16 繪圖日期 6/8

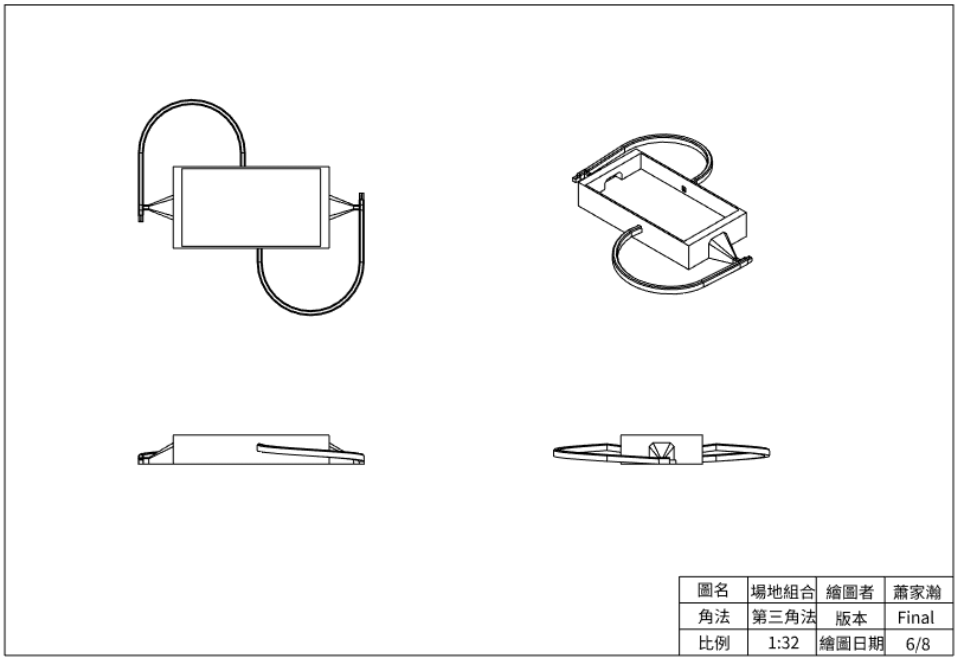
軌道


9.12
圖名 導球球門 繪圖者 蕭家瀚
角法 第三角法 版本 Final
比例 1:4 繪圖日期 6/8

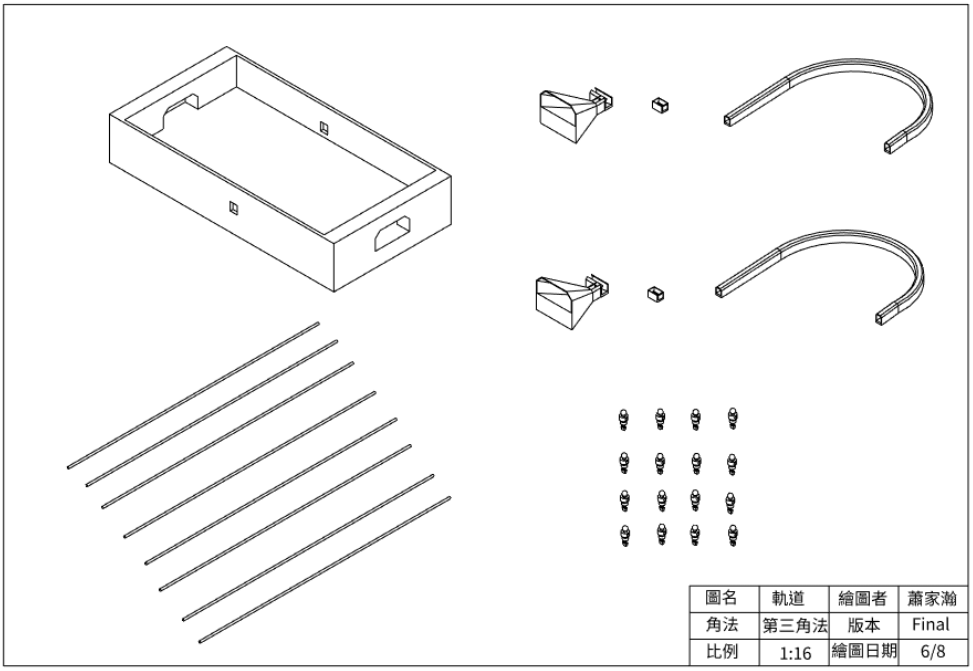
導球球門


1.5
2.362
1
1.2
圖名 擊球系統 繪圖者 蕭家瀚
角法 第三角法 版本 Final
比例 1:1 繪圖日期 6/8

擊球系統


圖名 場地組合 繪圖者 蕭家瀚
角法 第三角法 版本 Final
比例 1:32 繪圖日期 6/8

場地組合


圖名 軌道 繪圖者 蕭家瀚
角法 第三角法 版本 Final
比例 1:16 繪圖日期 6/8

組合圖 BOM

第四章 V-rep 動態模擬

4.1 送球機構設計與模擬

由於當初設計不量沒考慮周到導致距離太遠及斜度太過另外球含有很 0.75 的彈性係數，這導致球根本上不去有時還飛出軌道…

我想到的解決方法是加上蓋子以及在球於軌道中時寫個判斷式讓球傳送到離洞口較近的下坡，想不到球一開始就過不了洞，之後又寫了個讓球能滾出去的 code，這是偷吃步的方法也是利用程式輔助機構的方法，雖然很不真實但確實可行。

下方這段是個簡單的判別式: 當球的 X>0.905 同時 Y 小於-0.069 又同時 Z 小於 0.036 時，推球的機構往前 push，第二段也是同理只是相反邊，而第三段則是當球的 Z 方向數值大於 0.036 時兩者回到初始位置。

```
function reborn_ball_count_goal()

    if Ball_s[1] > 0.905 and Ball_s[2] < -0.069 and Ball_s[3]
        sim.setJointTargetVelocity(Push_1,1)
    end

    if Ball_s[1] < -1.15 and Ball_s[2] < -0.123 and Ball_s[3]
        sim.setJointTargetVelocity(Push_2,-1)
    end

    if Ball_s[3] > 0.036 then
        sim.setJointTargetVelocity(Push_1,-1)
        sim.setJointTargetVelocity(Push_2,1)
    end

end
```

Return_ball&score_1

下方的這幾行也是相同道理，當球的 XYZ 同時滿足條件時 (也是擊球後，球滾到一半時的位置) 被傳送到另一位置，在從另一位置傳送到另一位置。

```
function reborn_ball_count_goal()
    if Ball_s[1] < -1.156 and Ball_s[2] > 0.163 and Ball_s[3]
    sim.setObjectPosition(Ball,-1,{-0.20361, 0.2739, 0.15735}
    end
    if Ball_s[1] < -0.20361 and Ball_s[2] > 0.2740 and Ball_s
    sim.setObjectPosition(Ball,-1,{-0.2049, 0.23, 0.1482})
    end
    if Ball_s[1] > 0.9056 and Ball_s[2] <  -0.4247 and Ball_s
    sim.setObjectPosition(Ball,-1,{-0.043425,  -0.60401, 0.15
    end
    if Ball_s[1] > -0.04753 and Ball_s[2] <  -0.5240 and Ball_
    sim.setObjectPosition(Ball,-1,{-0.03667,  -0.48443, 0.148
    end
end
```

Return_ball&score_2

影片

檔　案: https://github.com/mdecadp2018/site-40623130/blob/gh-pages/v-rep/robot/C_Goal_4.3.ttt

4.2 系統功能展示

這裡紀載了全部的手足球版本以及詳細歷程 “可點擊”

1. 人機對決

最終版本的 “玩家與電腦” 對打

操作說明: 方向鍵的左右下按鍵、z、x、c、v 分別按順序從 z 的守門員到 v 的最前排、R 鍵用來讓球回位至正中央 (怕卡 bug)

影片

檔　　案: https://github.com/mdecadp2018/site-40623130/blob/gh-pages/v-rep/robot/FinalVersion.ttt

2. 機 ^{2} 對決

寫完人機對打後，花了不到 1 小時寫完電腦對電腦的版本

檔　　案: https://github.com/mdecadp2018/site-40623130/blob/gh-pages/v-rep/robot/FinalVersion_EX.ttt

3. 人形歸位

我將每根桿件設定成 “當按了按鍵變成擊球狀態後會到達邊界數值，如果到達邊界直做減速度運動回到初始狀態”，下方 code 中為 +dVel，其原因是因為原先設定擊球為做減速度運動而回球則加入加速度。

```
A= sim.getObjectHandle('RS1_PO1')
PA= sim.getObjectPosition(A,-1)
if PA[1] >= -0.712 then
    Roller_v = Roller_v + dVel*3
end
B= sim.getObjectHandle('RS2_PO1')
PB= sim.getObjectPosition(B,-1)
if PB[1] >= -0.531 then
    Roller_v2 = Roller_v2 + dVel*3
end
C= sim.getObjectHandle('RS4_PO2')
PC= sim.getObjectPosition(C,-1)
if PC[1] >= -0.197 then
    Roller_v3=Roller_v3 + dVel*5
end
D= sim.getObjectHandle('RS3_PO1')
PD= sim.getObjectPosition(D,-1)
if PD[1] >= 0.137 then
    Roller_v4=Roller_v4 + dVel*5
end
```

Return_position

檔　　案: https://github.com/mdecadp2018/site-40623130/blob/gh-pages/v-rep/robot/www2.ttt

4. 電腦多人形球位判斷式

將擊球的人行分成三個後，紀錄每兩個人型之間的距離，利用補正讓球穿過人形與人形之間的分界線時，判斷式 work，加入補正的數值讓最左或是最右邊的人形能對正球體。此程式為最初簡易版本。

影片

檔 案:https://github.com/mdecadp2018/site-40623130/blob/gh-pages/v-rep/robot/ww5.ttt

5. 回球判斷式-無軌道 (包含計分判斷式)

一開始還沒有導球軌道時，為了不用一直關掉在啟動，寫了一個簡單的回位 code，讓球經過球門時，由於球門有判斷式屬於邊界，一通過就會回到設定好的球。

而計分也是利用相通的方法，且由於此程式會自動形成迴圈所以讓他判斷成每當球經過時 +1 分後回傳。此程式為最初簡易版本。

以下都更改於function sysCall_actuation()中

```
        if 1 then  --C_goalkeeper
            if X <= 0.02 then
                sim.setJointTargetVelocity(LR1,-5)
            elseif X > 0.02 then
                sim.setJointTargetVelocity(LR1,5)
            end
        end
----------------
        if Ball_s[1] < -0.85 or Ball_s[1] >= 0.61 then
            sim.setObjectPosition(Ball,-1,{-0.1429, -0.1334, 0.5})
        end
```

Return_ball

影片

檔　　案: https://github.com/mdecadp2018/site-40623130/blob/gh-pages/v-rep/robot/Reborn.ttt

6. 偽-回球判斷式-有軌道

也就是送球機構於上一章節的 “送球機構設計與模擬” 討論過

影片

檔　案: https://github.com/mdecadp2018/site-40623130/blob/gh-pages/v-rep/robot/C_Goal_4.3.ttt

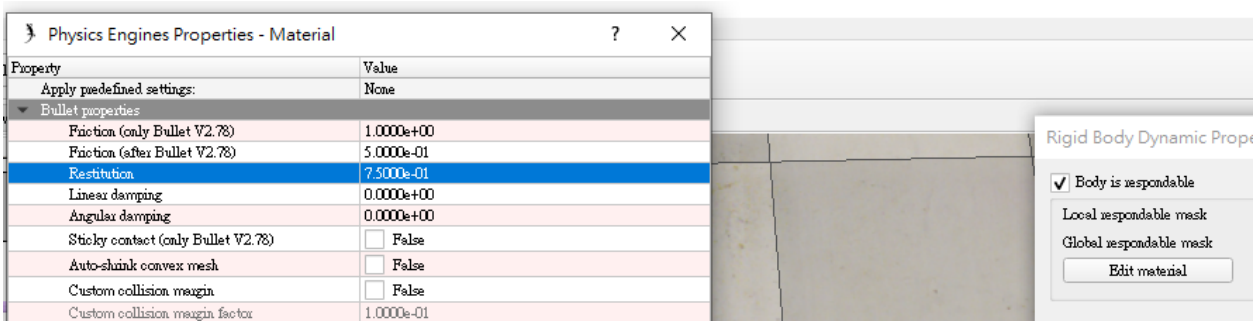
7. 問題與討論

Q1. 怎麼沒有雙打?

A1. 按鍵過多不好操控，最後會變成賽外賽?

Q2. 彈性係數怎麼調整?

A2. 彈性系數於球的材質中調整，而材質的調整在關係設定的下方，表中的 Restitution 就是了最大彈性為 1 最小為 0，越大越彈。



Restitution

Q3. 怎麼不用 python 寫??

A3. 我會選擇使用 Vrep 內嵌的 Lua 編寫的原因

優點:

python : 可執行運算值較大的編譯、可加外部其他的程式合用

lua : Vrep 內部沿用, 延遲現象較少、編譯很方便

缺點:

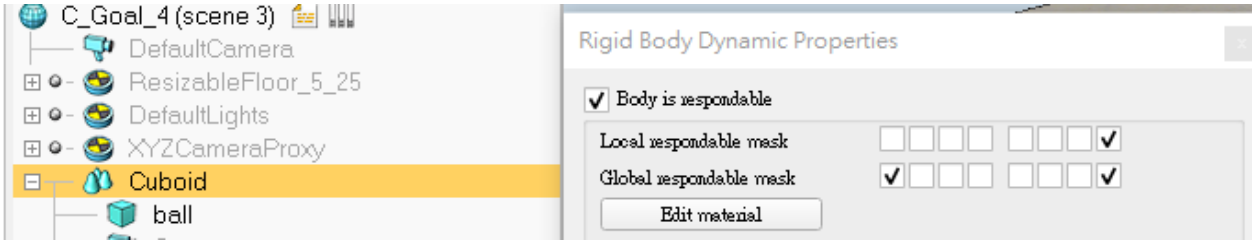
python : 會有爆 ping 問題導致延遲、產生很多衝突可能性高

lua : 太多運算時直接停止、內部函式限制多

目前體會到的優缺點大概就這樣，而且感覺我們班大部分的人都是用 python 寫，所以我想用少數人用的 lua。

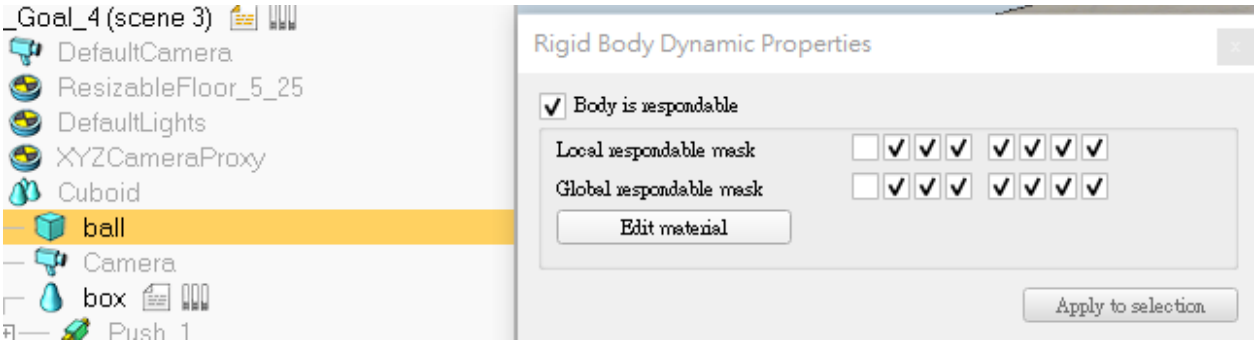
T4: 反作用設定: 包含場地、球、桿件、人形 ((解釋起來很麻煩但大概是這樣

第一個為場地: 第一行設定讓除了球、人形的桿件能反應但設定第八行讓球、人形可反應但桿件無法，其餘不勾選



Respondable_court

第二個為球: 第一行不勾選讓桿件與人行無法反應，但勾選其他行 (第八) 讓人形能反應

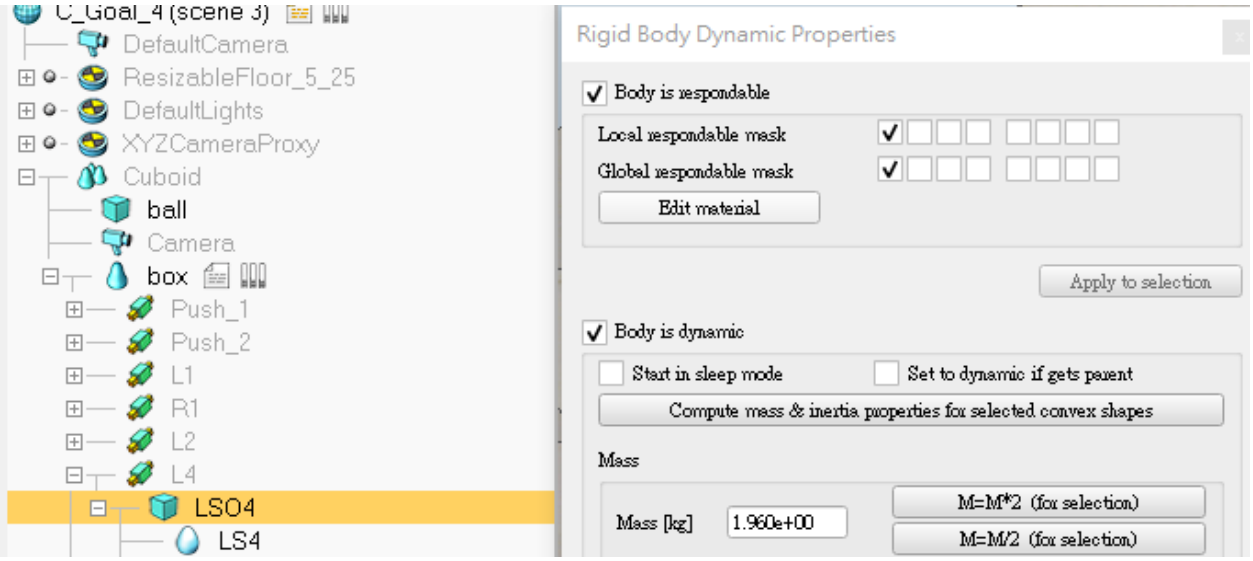


Respondable_ball

T5. 運行時建議先點一下烏龜太快會增加電腦能力也有機率 bug

Turtle

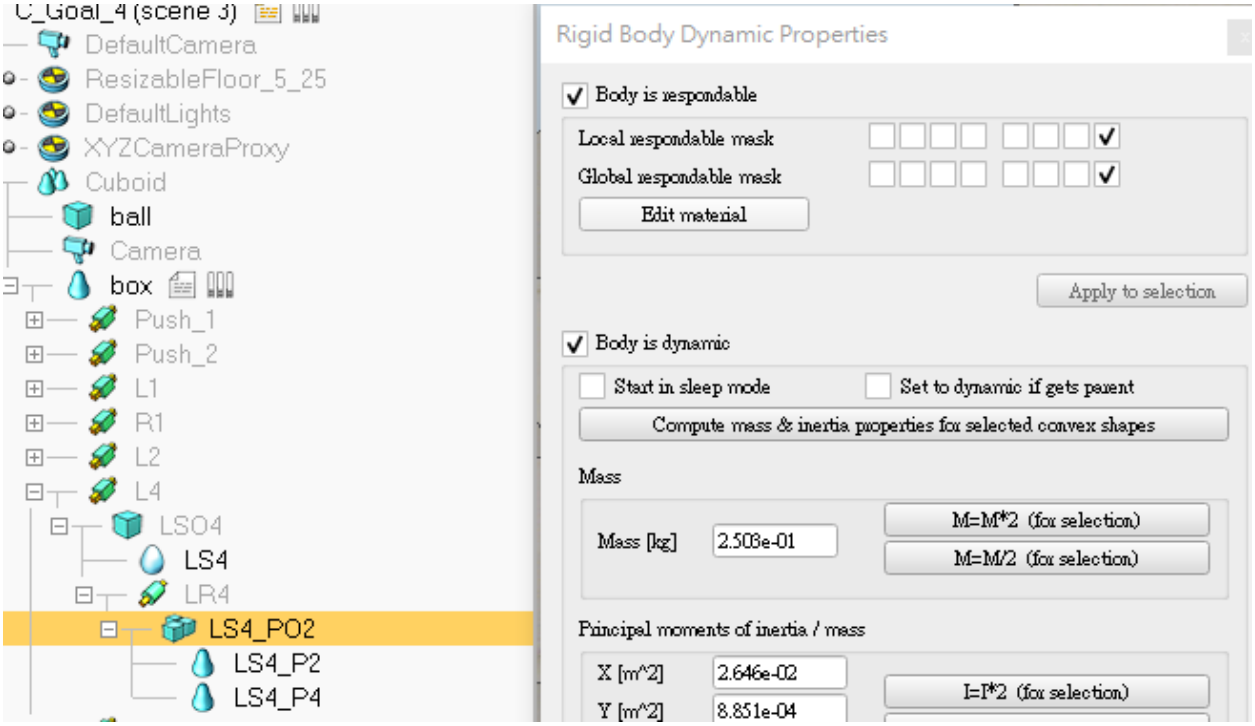
第三個為桿件: 只與場地反應 ((這裡注意，桿件本身與人型成父子屬性



Respondable_handle

最後是人形: 勾選第八行讓球反應可踢球且因與場地反應所以不出場地

理解這個需要一點邏輯…如果不懂就照勾選吧…



Respondable_doll

8.PDF 限定-Lua 解說

function sysCall_init() 中的基本上都是必要的參數輸入及物件設定

function sysCall_actuation() 則比較像是個無線循環的大型函式庫

```
function sysCall_init()
    Ball=sim.getObjectHandle('ball')
    --player--
    Slide_rail= sim.getObjectHandle('R1')
    Roller= sim.getObjectHandle('RR1')
    Slide_rail2= sim.getObjectHandle('R2')
    Roller2= sim.getObjectHandle('RR2')
    Slide_rail3= sim.getObjectHandle('R3')
    Roller3= sim.getObjectHandle('RR3')
    Slide_rail4= sim.getObjectHandle('R4')
    Roller4= sim.getObjectHandle('RR4')
    Slide_rail_v=0
    Roller_v=0
    Roller_v2=0
    Roller_v3=0
    Roller_v4=0
    dVel=0.5
    --computer--
    Body=sim.getObjectHandle('LSO1')
    Body2=sim.getObjectHandle('LSO2')
    Body3=sim.getObjectHandle('LSO3')
    Body4=sim.getObjectHandle('LSO4')
    LR1=sim.getObjectHandle('LR1')
    LR2=sim.getObjectHandle('LR2')
    LR3=sim.getObjectHandle('LL3')
    LR4=sim.getObjectHandle('LR4')
    L1=sim.getObjectHandle('L1')
    L2=sim.getObjectHandle('L2')
    L3=sim.getObjectHandle('L3')
    L4=sim.getObjectHandle('L4')
    --goal--
    Player_G = 0
    Computer_G = 0
    --reball--
    Push_1=sim.getObjectHandle('Push_1')
    Push_2=sim.getObjectHandle('Push_2')
end
```

Final-1

裡頭標註’player’ 為玩家的操作 code 設定 player 下方的幾行屬於玩家操作人形的自動回初始位置 code，

```
function sysCall_actuation()
    --player--
    A= sim.getObjectHandle('RS1_PO1')
    PA= sim.getObjectPosition(A,-1)
    if PA[1] >= -0.712 then
        Roller_v = Roller_v + dVel*2.5
    end
    B= sim.getObjectHandle('RS2_PO1')
    PB= sim.getObjectPosition(B,-1)
    if PB[1] >= -0.531 then
        Roller_v2 = Roller_v2 + dVel*5
    end
    C= sim.getObjectHandle('RS4_PO2')
    PC= sim.getObjectPosition(C,-1)
    if PC[1] >= -0.197 then
        Roller_v3=Roller_v3 + dVel*5
    end
    D= sim.getObjectHandle('RS3_PO1')
    PD= sim.getObjectPosition(D,-1)
    if PD[1] >= 0.137 then
        Roller_v4=Roller_v4 + dVel*5
    end
```

Final-2

如圖所示，這是玩家用 keyborad 的設定有 r、z、x、c、v 下張圖還有 ←↓→，這個 while message 不等於-1 為條件時執行，這不是一直輸出的 code 他會等待輸入源再用判斷句控制想要的功能，裡頭鍵盤的數字屬於 {Unicode} 的設定。

```
message,auxiliaryData=sim.getSimulatorMessage()
while message~=-1 do
    if (message==sim.message_keypress) then
        if (auxiliaryData[1]==114) then   --"r"--restart
         sim.setObjectPosition(Ball,-1,{-0.15, -0.1334, 0.5})
        end
        if (auxiliaryData[1]==122) then  --"z"--goalkeeper
          Roller_v = Roller_v - dVel*60
          if (Roller_v > 0) then
            Roller_v = 0
          end
        end
        if (auxiliaryData[1]==120) then   --"x"--2
          Roller_v2 = Roller_v2 - dVel*15
          if (Roller_v2 > 0) then
            Roller_v2 = 0
          end
        end
        if (auxiliaryData[1]==99) then  --"c"--3
          Roller_v3 = Roller_v3 - dVel*15
          if (Roller_v3 > 0) then
            Roller_v3 = 0
          end
        end
        if (auxiliaryData[1]==118) then  --"v"--4
          Roller_v4 = Roller_v4 - dVel*15
          if (Roller_v4 > 0) then
            Roller_v4 = 0
          end
        end
```

Final-3

除了 R 可以讓球回原點外，ZXCV 還有 ←↓→ 都屬於增加及減少加速度的函式，但 ↓(2008) 比較算是排錯的按鍵跟 R 很像，當人形卡在一半時可用，而其中還有四個小型判斷句，是用來讓操作更順暢而編入地，最下方那行 message 對應到上一張圖最上方的 code。

```
        if (auxiliaryData[1]==2008) then --"down"--return_player
            Roller_v = Roller_v + dVel*20
            Roller_v2 = Roller_v2 + dVel*20
            Roller_v3 = Roller_v3 + dVel*20
            Roller_v4 = Roller_v4 + dVel*20
            if (Roller_v < 0) then
                Roller_v = 0
            end
            if (Roller_v2 < 0) then
                Roller_v2 = 0
            end
            if (Roller_v3 < 0) then
                Roller_v3 = 0
            end
            if (Roller_v4 < 0) then
                Roller_v4 = 0
            end
        end
        if (auxiliaryData[1]==2009) then --"right"
            Slide_rail_v = Slide_rail_v + dVel/5
            if (Slide_rail_v < 0) then
                Slide_rail_v = 0
            end
        end
        if (auxiliaryData[1]==2010) then --"left"
            Slide_rail_v = Slide_rail_v - dVel/5
            if (Slide_rail_v > 0) then
                Slide_rail_v = 0
            end
        end
    end
    message,auxiliaryData=sim.getSimulatorMessage()
end
```

Final-4

如圖所示，是四軸同動但不同數度的設定。

```
Slide_rail_v2 = Slide_rail_v/1.3
Slide_rail_v3 = Slide_rail_v/1.5
Slide_rail_v4 = Slide_rail_v/1.4
sim.setJointTargetVelocity(Slide_rail,Slide_rail_v)
sim.setJointTargetVelocity(Roller,Roller_v)
sim.setJointTargetVelocity(Slide_rail2,Slide_rail_v2)
sim.setJointTargetVelocity(Roller2,Roller_v2)
sim.setJointTargetVelocity(Slide_rail3,Slide_rail_v3)
sim.setJointTargetVelocity(Roller3,Roller_v4)
sim.setJointTargetVelocity(Slide_rail4,Slide_rail_v4)
sim.setJointTargetVelocity(Roller4,Roller_v3)
```

Final-5

此為電腦操作的程式設定，第一行是 get Body(人形) 的 XYZ 數值 code，第二行則是球，圖中 XY 各指人形-球得到的 X 及 Y 值，下方就是判斷多少距離下哪個桿件旋轉並擊球，YYYYY 是之後設定於左右移動的速度越快越容易打到球，由於只有一個人型比較好懂。

```
--computer--
Body_s = sim.getObjectPosition(Body,-1)
Ball_s = sim.getObjectPosition(Ball,-1)
X = Body_s[1] - Ball_s[1]
Y = Body_s[2] - Ball_s[2]
if 1 then                          --C_goalkeeper
    if X <= 0.02 then
        sim.setJointTargetVelocity(LR1,-5)
        sim.setJointTargetVelocity(L1,0)
    elseif X > 0.02 or X <= 0.01 then
        sim.setJointTargetVelocity(LR1,5)
        YYYYY = Y*7
        sim.setJointTargetVelocity(L1,YYYYY)
    end
end
```

Final-6

接下是兩個人形的換位判定，假設中間有個-0.133 的牆，大於或小於時由於補正值的關係會剛好被較靠近球的左右判定拉動兩個人形，所以到中間時會瞬間換位，裏頭的 C-K-B-X2&3&4() 的函式被我拉到 function sysCall_actuation() 的外面看起來比較整齊…

```
Body_s2 = sim.getObjectPosition(Body2,-1)
X2_1 = Body_s2[1] - Ball_s[1]
Y2_1 = Body_s2[2] - Ball_s[2] - 0.15
Y2_2 = Body_s2[2] - Ball_s[2] + 0.15
if X2_1 >= 0.01 then            --C_second--
    if Ball_s[2] <= -0.133 then
        C_K_B_X2()
    Y2_1 = Y2_1*5
    sim.setJointTargetVelocity(L2,Y2_1)
    elseif Ball_s[2] > -0.133 then
        C_K_B_X2()
    Y2_2 = Y2_2*5
    sim.setJointTargetVelocity(L2,Y2_2)
    end
end
```

```
Body_s4 = sim.getObjectPosition(Body4,-1)
X4_1 = Body_s4[1] - Ball_s[1]
Y4_1 = Body_s4[2] - Ball_s[2] - 0.15
Y4_2 = Body_s4[2] - Ball_s[2] + 0.15
if X4_1 >= 0.01 then              --C_third--
    if Ball_s[2] <= -0.133 then
        C_K_B_X4()
    Y4_1 = Y4_1*3
    sim.setJointTargetVelocity(L4,Y4_1)
    elseif Ball_s[2] > -0.133 then
        C_K_B_X4()
    Y4_2 = Y4_2*3
    sim.setJointTargetVelocity(L4,Y4_2)
    end
end
```

Final-7

也是依距離判定是否旋轉擊球。桿件 3 則是三判斷的換位判定，了解原理後，其實都差不多。

```
Body_s3 = sim.getObjectPosition(Body3,-1)
X3_1 = Body_s3[1] - Ball_s[1]
Y3_1 = Body_s3[2] - Ball_s[2]
Y3_2 = Body_s3[2] - Ball_s[2] +0.2487
Y3_3 = Body_s3[2] - Ball_s[2] -0.2487
if X3_1 >= 0.01 then                --C_fourth--
    if Ball_s[2] < -0.0028 and Ball_s[2] >= -0.2487 then
        C_K_B_X3()
    Y3_1 = Y3_1*2.5
    sim.setJointTargetVelocity(L3,Y3_1)
    elseif Ball_s[2] >= -0.0028 then
        C_K_B_X3()
    Y3_2 = Y3_2*2.5
    sim.setJointTargetVelocity(L3,Y3_2)
    elseif Ball_s[2] < -0.2487 then
        C_K_B_X3()
    Y3_3 = Y3_3*2.5
    sim.setJointTargetVelocity(L3,Y3_3)
    end
end
```

Final-8

reborn-ball-count-goal() 也是外拉的函式集，與下方的 C-K-B-X2&3&4() 一樣，但裏頭的編譯很不一樣…

```
    --reborn_ball & count goal--
    reborn_ball_count_goal()
end

function C_K_B_X2() --C_second_kick--
    if X2_1 < 0.06 then
    sim.setJointTargetVelocity(LR2,-5)
    elseif X2_1 > 0.03 then
    sim.setJointTargetVelocity(LR2,5)
    end
end

function C_K_B_X3() --C_third_kcik--
    if X3_1 < 0.06 then
    sim.setJointTargetVelocity(LR3,-5)
    elseif X3_1 > 0.03 then
    sim.setJointTargetVelocity(LR3,5)
    end
end

function C_K_B_X4() --C_fourth_kick--
    if X4_1 < 0.06 then
    sim.setJointTargetVelocity(LR4,-5)
    elseif X4_1 > 0.03 then
    sim.setJointTargetVelocity(LR4,5)
    end
end
```

Final-9

此圖已於 “送球機構設計與模擬” 的章節中說明。

人機對決大概就這樣，而電腦間的對決由於設定同理就不說明了。

```
function reborn_ball_count_goal()
    if Ball_s[1] < -0.85 and Ball_s[1] > -0.87 then
        Player_G = Player_G + 1
    end
    if Ball_s[1] > 0.61 and Ball_s[1] < 0.62 then
        Computer_G = Computer_G + 1
    end
    if Ball_s[1] < -0.85 or Ball_s[1] > 0.61 then
        print[[Player : Computer]]
        print('      '..Computer_G..'     '..Player_G)
    end

    if Ball_s[1] > 0.905 and Ball_s[2] < -0.069 and Ball_s[3] < 0.036 then
        sim.setJointTargetVelocity(Push_1,1)
    elseif Ball_s[3] > 0.036 then
        sim.setJointTargetVelocity(Push_1,-1)
    end
    if Ball_s[1] < -1.15 and Ball_s[2] < -0.123 and Ball_s[3] < 0.036 then
        sim.setJointTargetVelocity(Push_2,-1)
    elseif Ball_s[3] > 0.036 then
        sim.setJointTargetVelocity(Push_2,1)
    end

    if Ball_s[1] < -1.156 and Ball_s[2] > 0.163 and Ball_s[3] > 0.058 then
    sim.setObjectPosition(Ball,-1,{-0.20361, 0.2739, 0.15735})
    end
    if Ball_s[1] < -0.20361 and Ball_s[2] > 0.2740 and Ball_s[3] > 0.15735 then
    sim.setObjectPosition(Ball,-1,{-0.2049, 0.23, 0.1482})
    end
    if Ball_s[1] > 0.9056 and Ball_s[2] <  -0.4247 and Ball_s[3] > 0.0592 then
    sim.setObjectPosition(Ball,-1,{-0.043425,  -0.60401, 0.15735})
    end
    if Ball_s[1] > -0.04753 and Ball_s[2] <  -0.5240 and Ball_s[3] > 0.15735 then
    sim.setObjectPosition(Ball,-1,{-0.03667,  -0.48443, 0.1482})
    end
```

Final-10

第五章 參考文獻