

國立虎尾科技大學

機械設計工程系

cd2023 2a-pj2ag5 分組報告

網際足球泡泡機器人場景設計

Web-based bubbleRob Football  
Scene Design

指導教授：嚴家銘老師

班級：四設二甲

學生：李凱新(41023106)

王翔楷(41023113)

李學淵(41023125)

張昱棠(41023153)

中華民國

112年5月

## 摘要

由於矩陣計算、自動求導技術、開源開發環境、多核 GPU 運算硬體等這四大發展趨勢，促使 AI 領域快速發展，藉由這樣的契機，將實體機電系統透過虛擬化訓練提高訓練效率，再將訓練完的模型應用到實體上。

此專案是 w3 作業所做的泡泡機器人的延伸，繪製機器人後導入 CoppeliaSim 模擬環境並給予對應設置，使用 zmqRemoteAPI 與同組組員協同控制 bubbleRob，在我們所建立的場景內踢球競賽，並同時加入記分板顯示場上比分狀態。

## Abstract

Due to the four major development trends of multidimensional arrays computing, automatic differentiation, open source development environment, and multi-core GPUs computing hardware. The rapid development of the AI field has been promoted. In view of this development, the physical mechatronic systems can gain machine learning efficiency through their simulated virtual system training process. And afterwards to apply the trained model into real mechatronic systems.

This project is an extension of the bubble robot created for the w3 assignment. After designing the robot, we integrated it into the CoppeliaSim simulation environment and configured the necessary settings. We used zmqRemoteAPI to collaboratively control the bubbleRob with our team members. In the scene we created, we had a soccer competition where the robots played against each other, and we also added a scoreboard to display the current score on the field.

## 誌 謝

在此鄭重感謝製作以及協助本分組報告完成的所有人員，共同討論遇上的問題並思考解決方案，首先向嚴家銘老師致謝，不厭其煩的回答我們的提問，總是像燈塔一樣為我們指引出最正確的方向。最後是由本分組組員同心協力才得以完成本報告，特此感謝。

## 目 錄

摘要.....	i
Abstract .....	i
誌 謝.....	i
第一章 前言 .....	1
1.1 規則.....	1
第二章 場景建立 .....	2
2.1 匯入球場及球員 .....	2
2.2 記分板建立.....	3
第三章 程式講解 .....	6
3.1 變色程式講解 .....	6
3.2 機械式程式講解 .....	11
第四章 2a2-pj2ag5 製作心得 .....	12
4.1 李凱新心得.....	12
4.2 王翔楷心得.....	12
4.3 李學淵心得.....	12
4.4 張昱棠心得.....	12
第五章 完成作業 .....	13

## 圖 目 錄

圖 2.1 匯入球場 stl 檔 . . . . .	2
圖 2.2 改變球員顏色 . . . . .	2
圖 2.3 測試用記分板 . . . . .	3
圖 2.4 第二版記分板 . . . . .	3
圖 2.5 變色顯示得分 . . . . .	4
圖 2.6 第四版記分板 . . . . .	4
圖 2.7 第四版記分板背面 . . . . .	5
圖 2.8 第五版記分板 . . . . .	5
圖 2.9 第五版記分板原理 . . . . .	5
圖 3.1 改變顏色程式 1 . . . . .	6
圖 3.2 改變顏色程式 2 . . . . .	6
圖 3.3 改變顏色程式 3 . . . . .	7
圖 3.4 改變顏色程式 4 . . . . .	7
圖 3.5 改變顏色程式 5 . . . . .	8
圖 3.6 改變顏色程式 6 . . . . .	8
圖 3.7 改變顏色程式 7 . . . . .	9
圖 3.8 改變顏色程式 8 . . . . .	9
圖 3.9 改變顏色程式 9 . . . . .	10
圖 3.10 改變顏色程式 10 . . . . .	10

圖 3.11 機械式程式	11
圖 3.12 機械式程式 2	11
圖 5.1 完成作業	13

# 第一章 前言

## 1.1 規則

遊戲規則如下：

1. 球打入敵方即得一分。
2. 時間內進球數多的一方獲勝。

## 第二章 場景建立

## 2.1 匯入球場及球員

File-Import-Mesh，選擇要匯入的檔案匯入球場，如(圖.2.1)。

接著依序匯入球員及球，並且作球員顏色的更動，如(圖.2.2)。

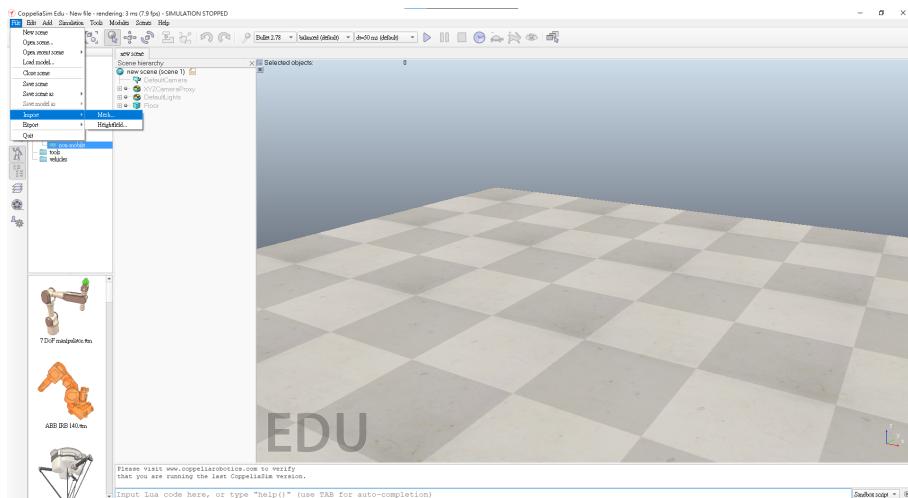


圖. 2.1: 匯入球場 stl 檔

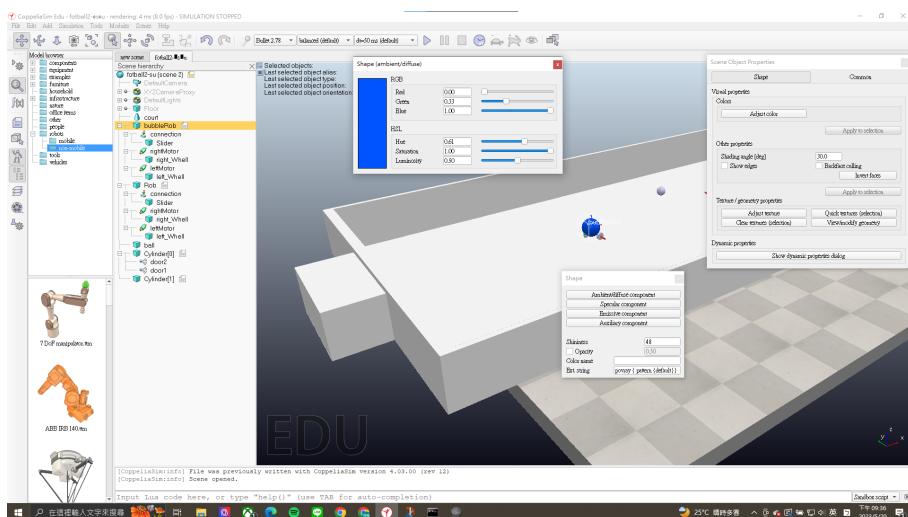


圖. 2.2: 改變球員顏色

變色方法：點選本體旁邊圖示-Adjust color-Ambient/diffuse component-拉動 RGB 調整顏色即可。

## 2.2 記分板建立

建立第一版記分板做為測試用途。

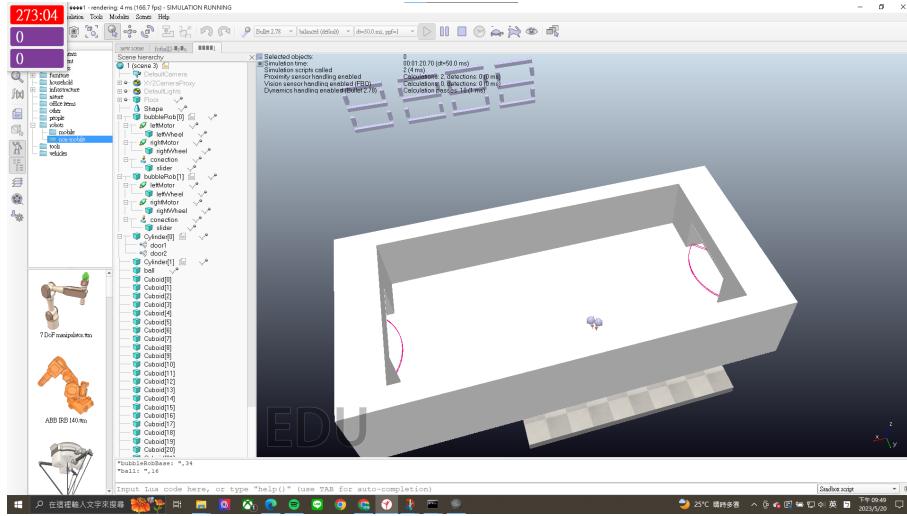


圖. 2.3: 測試用記分板

接著我們使用 Onshape 繪製了第二版記分板，匯入後進行爆炸拆件；步驟為 Edit-Gourping-Divide selected shape。因為我們是使用變換物件顏色來顯示得分數字，所以物件導入後的拆件動作件特別重要。

但由於第二版記分板，無法在 CoppeliaSim 爆炸成個別零件，無法達成我們想改變計分板顏色來實現計分功能的計畫，因此沒有採用。

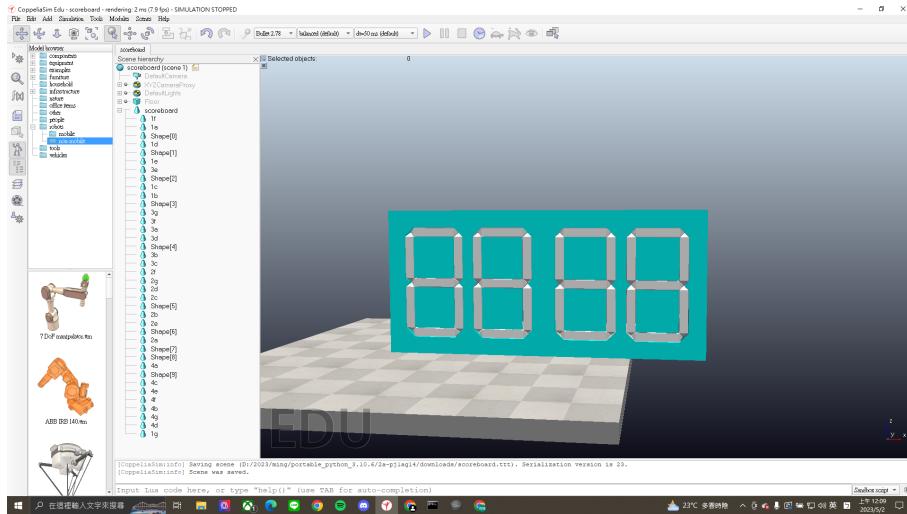


圖. 2.4: 第二版記分板

建立第三版記分板，匯入後成功拆件，也順利完成程式控制變色功能。如(圖.2.5)。

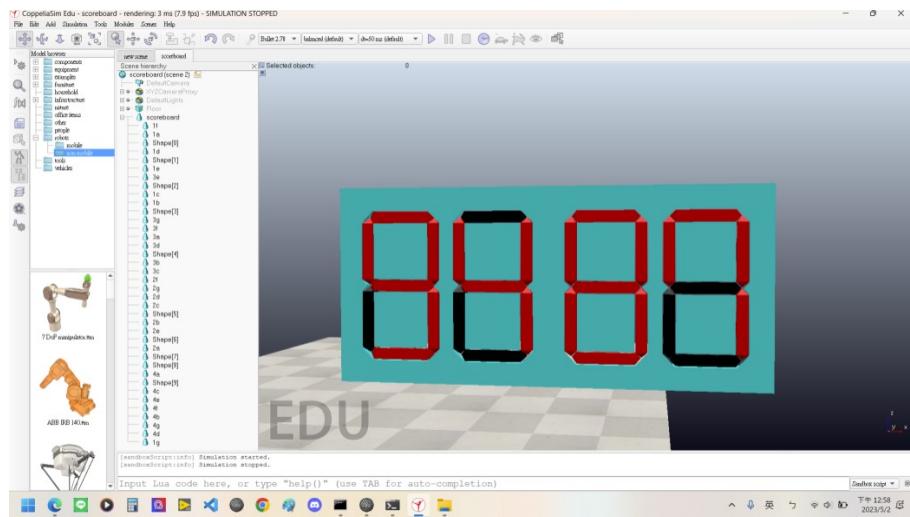


圖. 2.5: 變色顯示得分

第四版記分板建立，因前面對老師的要求理解錯誤，我們做成隨得分改變顏色的記分板設計，因此做了這版來滿足老師所要求的機械式設計。如(圖.2.7)紅色圓形部分為固定銷，白色圓形部分是可向前推動的銷，可實現將桿件向前推送達成數字顯示的效果。

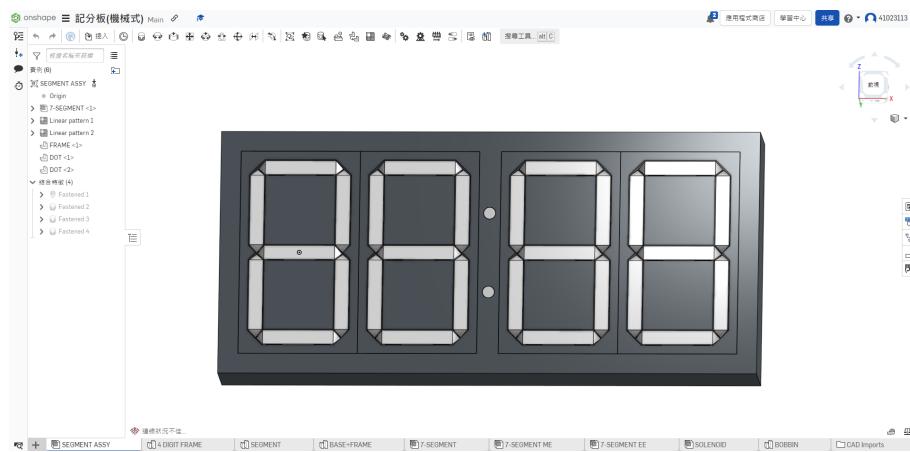


圖. 2.6: 第四版記分板



圖. 2.7: 第四版記分板背面

第五版記分板建立，第四版在經過組內討論後，發現顯示效果不容易判讀，因此建立第五版記分板如（圖.2.8），原理大致上與第四版相同，皆是使用 joint 推動顯示數字如（圖.2.9）。

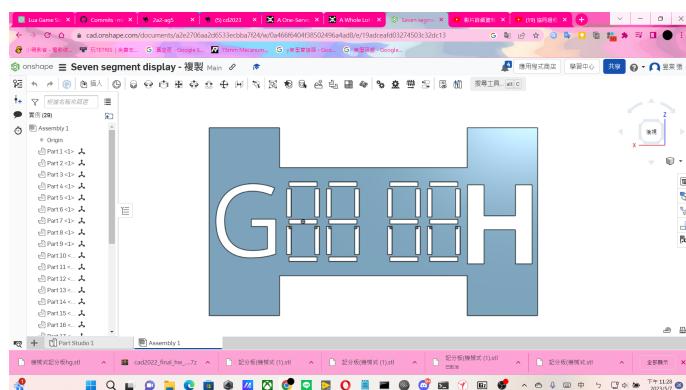


圖. 2.8: 第五版記分板

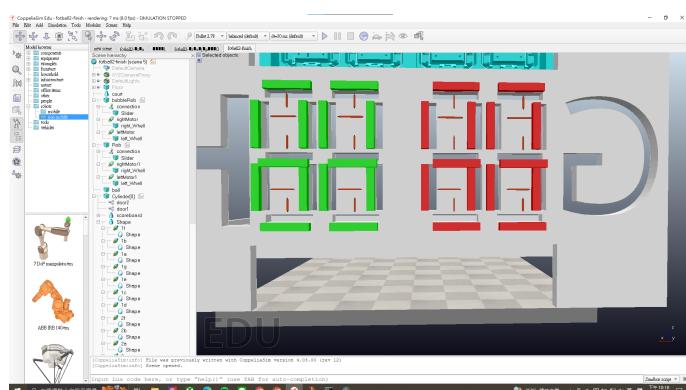


圖. 2.9: 第五版記分板原理

## 第三章 程式講解

### 3.1 變色程式講解

```
1 | function randomNumber()
2 |   math.randomseed(os.time())
3 |   return {tonumber(math.random(-2, 2) .. '.' .. math.random(0,9)),tonumber(math.random(-1, 1) .. '.' .. math.random(0,4))} 
4 | end
```

圖. 3.1: 改變顏色程式 1

這段程式碼，如 (圖.3.1)，定義了一個名為 randomNumber 的函數，當被呼叫時，它會產生一個由三個數字組成的列表 list，這三個數字是從指定範圍中隨機選擇而來的。在此函數中，首先使用 math.randomseed os.time 函數設置一個隨機數種子，以保證每次呼叫 randomNumber 函數時，產生的隨機數是不同的。接下來，使用 math.random -2, 2 函數從-2 到 2 之間選擇一個整數，並使用 math.random 0,9 函數從 0 到 9 之間選擇一個整數，這兩個整數組合起來成為一個小數，表示為第一個數字。同樣地，使用 math.random -1, 1 函數從-1 到 1 之間選擇一個整數，並使用 math.random 0,4 函數從 0 到 4 之間選擇一個整數，這兩個整數組合起來成為另一個小數，表示為第二個數字。最後，固定設置第三個數字為 1.0，並將這三個數字放入一個列表中，作為函數的返回值。

```
12 | function regress()
13 |   sim.pauseSimulation()
14 |   sim.setObjectPosition(bubbleRobBase, -1, initialBubbleRobPosition)
15 |   sim.setObjectOrientation(bubbleRobBase, -1, initialBubbleRobOrientation)
16 |   sim.setObjectPosition(ball, -1, randomNumber())
17 |   sim.setObjectOrientation(ball, -1, initialballOrientation)
18 |   sim.setObjectPosition(23, -1, initial)
19 |   sim.setObjectOrientation(23, -1, initial)
20 | end
```

圖. 3.2: 改變顏色程式 2

這段程式碼，如 (圖.3.2)，包含一個名為 regress 的函數，用於將機器人和球體重置到遊戲開始時的位置。在函數內部，程式暫停模擬運行，並使用 sim.setObjectPosition 和 sim.setObjectOrientation 函數將機器人和球體移回初始位置。

```

24 |     function Toclear()
25 |         for i = 1, 7 do
26 |             for x = 1, 4 do
27 |                 handle = handles[x][i]
28 |                 sim.setShapeColor(handle, nil, sim.colorcomponent_ambient_diffuse, color[2])
29 |             end
30 |         end
31 |

```

圖. 3.3: 改變顏色程式 3

如 (圖.3.3)Toclear 的函數，它的作用是清除計分板，也就是把所有的計分板 LED 燈恢復到預設的顏色。函數使用了巢狀的 for 迴圈，首先從 1 到 7 執行一遍，然後再從 1 到 4 執行一遍，這樣就可以執行所有的計分板 LED 燈。在迴圈內部，函數通過 handle 變量獲取每個 LED 燈的控制句柄，然後使用 sim.setShapeColor 函數將其顏色設置為預設顏色在 color 表中的索引為 2 的顏色。這樣，所有的計分板 LED 燈都會被恢復到預設的顏色，從而達到清除計分板的目的。

這段程式碼，如 (圖.3.4) 是一個顯示數字的函數，它有兩個參數，

```

39 |     function Number(displayNumber,ser)
40 |         for i = 1, 7 do
41 |             for j = 1, #specialNumbers[ser] do
42 |                 if i == specialNumbers[ser][j] then
43 |                     sim.setShapeColor(handles[displayNumber][i], nil, sim.colorcomponent_ambient_diffuse, color[1])
44 |                     break
45 |                 end
46 |             end
47 |         end
48 |

```

圖. 3.4: 改變顏色程式 4

一個是要顯示的數字 displayNumber，另一個是顯示器的類型 ser。這個函數的作用是將 displayNumber 這個數字顯示在屏幕上，屏幕的類型由 ser 決定。程式碼使用了嵌套的迴圈，第一個迴圈從 1 到 7 遍歷了七個數碼的 LED 顯示燈，第二個迴圈從 1 到 specialNumbers ser 的長度遍歷了指定類型的顯示屏上的特殊數碼。對於每個數碼燈，它會檢查這個燈是否是指定類型的顯示器的數字。如果是，它就會將該燈的顏色設置為顯示數字的顏色。

```

58 |     function scoreboard(number)
59 |         local numberString = tostring(number)
60 |         if #numberString < 2 then
61 |             numberString = '0' .. numberString
62 |         end
63 |         local tensDigit = tonumber(numberString:sub(1, 1))
64 |         local onesDigit = tonumber(numberString:sub(2, 2))
65 |         return{tensDigit,onesDigit}
66 |
67 --
```

圖. 3.5: 改變顏色程式 5

如 (圖.3.5) 這段程式碼定義了一個名為 scoreboard 的函數，該函數接受一個整數 number 作為參數，並將其轉換為兩位數的字串表示。如果傳入的 number 參數的長度小於 2，則會在數字前面添加一個 0，這樣就可以確保數字的表示總是兩位數。然後，這個函數會提取這個兩位數字串的每一位數字，分別存儲在一個名為 tensDigit 的變數和一個名為 onesDigit 的變數中。

```

72 |     function sysCall_init()
73 |         sensor1 = sim.getObject('./door1')
74 |         sensor2 = sim.getObject('./door2')
75 |         bubbleRobBase = 16
76 |         ball = 30
77 |         initialBubbleRobPosition = sim.getObjectPosition(bubbleRobBase, -1)
78 |         initialBubbleRobOrientation = sim.getObjectOrientation(bubbleRobBase, -1)
79 |         initia = sim.getObjectPosition(23, -1)
80 |         initial = sim.getObjectOrientation(23, -1)
81 |         initialBallPosition = sim.getObjectPosition(ball, -1)
82 |         initialBallOrientation = sim.getObjectOrientation(ball, -1)
83 |         -- do some initialization here
84 |         count = 18000 -- ??30?????????????
85 |         score1 = 0 -- ??????
86 |         score2 = 0
87 |
88 |         xml = [[
89 |             <ui closeable="false" resizable="false" activate="false">
90 |                 <label text="30:00" style="" (background-color: #F00; color: #FFF; font-size: 32px; font-weight: bold; padding: 10px)>
91 |                 <label text="0" style="" (background-color: #071E4B; color: #FFF; font-size: 32px; font-weight: bold; padding: 10px)>
92 |                 <label text="0" style="" (background-color: #071E4B; color: #FFF; font-size: 32px; font-weight: bold; padding: 10px)>
93 |             </ui>
94 |         ]]
95 |         ui = simUI.create(xml)
96 |         simUI.setPosition(ui, 0, 0, true)
97 |         for i = 1, 7 do
98 |             for x = 1, 4 do
99 |                 ii = tostring(x).. serialNumber[i]
100 |                 local handle = sim.getObjectHandle("./scoreboard/".. ii )
101 |                 handles[x][i] = handle
102 |             end
103 |         end
104 |     end

```

圖. 3.6: 改變顏色程式 6

如 (圖.3.6)，透過 sim.getObject 取得門的物件句柄，並存入 sensor1 和 sensor2 變數中。指定 bubbleRobBase 和 ball 變數分別為 16 和 30，代表這些物件在模擬場景中的物件句柄。透過 sim.getObjectPosition 和 sim.getObjectOrientation 取得模擬場景中物件的位置和方向資訊，分別存入 initialBubbleRobPosition、initialBubbleRobOrientation、initialballPosition 和 initialBallOrientation 變數中。將 23 號物件的位置和方向資訊分別存入 initia 和 initial 變數中。將 count、score1 和 score2 分別初始化為 18000、0 和 0。創建一個簡單的 UI 介面，包含三個標籤 label 元件，

分別代表倒數計時、兩個隊伍的得分。將介面移動到視窗左上角，並將其設為不可關閉、不可縮放和不可激活。設定用於記分牌的物件的物件句柄並存儲在 handles 二維陣列中。透過一個雙重迴圈，將所有的記分牌物件句柄存儲到 handles 變數中 andles 是一個二維陣列，用於儲存多個物體在仿真環境中的句柄 handle。二維陣列意味著它包含多個一維陣列，每個一維陣列都儲存了某一個維度上的物體句柄。在這裡，handles 有四個一維陣列，分別儲存了四個不同的元件在七個不同位置的句柄。通過這個二維陣列，程式可以方便地訪問並修改這些物體的屬性。

```

119 | function sysCall_actuation()
120 |     result1=sim.readProximitySensor(sim.getObject('./door1'))
121 |     result2=sim.readProximitySensor(sim.getObject('./door2'))
122 |     -- 0 or 1
123 |     if(result1>0)then
124 |         regress()
125 |         score1 = score1 +1
126 |
127 | end

```

圖. 3.7: 改變顏色程式 7

如 (圖.3.7)，此函數為一個回調函數，它會在仿真器每個時間步驟中被自動調用。在此函數中，首先透過 sim.getObject 'door1' 和 sim.getObject 'door2' 獲取到與感測器關聯的對象。然後通過 sim.readProximitySensor 函數檢測與感測器對應的物體是否被觸發，並將檢測結果存儲在 result1 和 result2 變量中。如果 result1 大於 0，表示感測器檢測到物體，此時會調用 regress 函數和加分操作，即分數 score1 加 1。

```

133 | if(result2>0)then
134 |     regress()
135 |     score2 = score2 +1
136 |
137 | end

```

圖. 3.8: 改變顏色程式 8

如 (圖.3.8)，如果 result2 大於 0，表示感測器檢測到物體，此時會調用 regress 函數和加分操作，即分數 score2 加 1。

```

139 if count > 0 then
140     count = count - 1
141     local minutes = math.floor(count / 60)
142     local seconds = count % 60
143     local timeStr = string.format("%d:%02d", minutes, seconds)
144     simUI.setLabelText(ui, 10, timeStr)
145     simUI.setLabelText(ui, 20, tostring(score1))
146     simUI.setLabelText(ui, 30, tostring(score2))
147     red = scoreboard(score1)
148     blue = scoreboard(score2)
149     Toclear()
150     for i = 1, 2 do
151         Number(i,red[i])
152         Number(i+2,blue[i])
153     end
154 else
155     sim.stopSimulation()
156 end
157
158 end

```

圖. 3.9: 改變顏色程式 9

如(圖.3.9)，此段程式碼為在仿真環境中，用來更新計分板和倒數計時器的功能。當遊戲倒數計時器仍有剩餘時間 count 大於 0，會依照每個計數間隔 1 秒進行更新。更新內容包含：剩餘時間的計算、將計算後的時間顯示在 UI 的倒數計時器上、顯示當前紅藍雙方的得分，同樣也是透過 UI 的 label 顯示。接著，程式會進入計分板的顯示功能。這邊透過呼叫 Toclear，先清除所有計分板的數字。再透過迴圈讀取紅藍雙方的分數，分別呼叫 Number 函數，在計分板上顯示對應的數字。若遊戲倒數計時器已歸零，則程式會執行 sim.stopSimulation，停止遊戲仿真。如

```

166 function sysCall_cleanup()
167     Toclear()
168 end

```

圖. 3.10: 改變顏色程式 10

(圖.3.10)，此程式為清理函數，當仿真停止時，會呼叫此函數，其目的為將目前顯示在分數板上的數字全部清除，使得下一次的遊戲能夠從零開始顯示分數。在此函數中，會呼叫之前已定義的 Toclear 函數，該函數會將所有的數字方塊改變為背景色，以達到清空的效果。

## 3.2 機械式程式講解

```
1 | joint=sim.getObjectHandle('./joint')
2 | sim.setJointTargetPosition(joint,0.1)
3 | --物體移動到絕對座標(0.1)
4 | sim.setJointTargetPosition(joint,0)
5 | --物體移動到絕對座標(0.0)
```

圖. 3.11: 機械式程式

如 (圖.3.11)，完成馬達操控。

```
1 | function sysCall_init()
2 |     joint=sim.getObjectHandle('./Cuboid[0]/joint')
3 |     sim.setJointTargetPosition(joint,-0.1)
4 | end
```

圖. 3.12: 機械式程式 2

如 (圖.3.12)，測試記分板顯示數字 0。

## 第四章 2a2-pj2ag5 製作心得

### 4.1 李凱新心得

這次四人的小組很感謝隊友們的強勁的能力，我們很快就有成果，會變色的計分板還有機械式計分板，我在隊伍裡面擔任統整全部人的結果，長時間都是在查程式是什麼意思，學習到很多東西，還學習到 onshape 更靈活應用，所以我很感謝隊友們。

### 4.2 王翔楷心得

這次的專案與上一次最大的不同是人數變多，分工也相對變得更細、更明確，用在前一個專案所學的技巧來對組內做貢獻，這次的協同也讓我更加熟悉了 zmq 的應用。

### 4.3 李學淵心得

這次 pj2 讓我學會了 zmqRemoteAPI 控 BubbleRob 並編寫了計時器的程式，對於 lua 的陣列的產生與取出有更深的了解，以及 python 與 lua 的差別有進一步的認識。

### 4.4 張昱棠心得

透過這次的專案，我深刻了解到如何規劃四人的協同專案，按時完成老師交付的進度，也讓我對 onshape 的使用更加精進。

## 第五章 完成作業

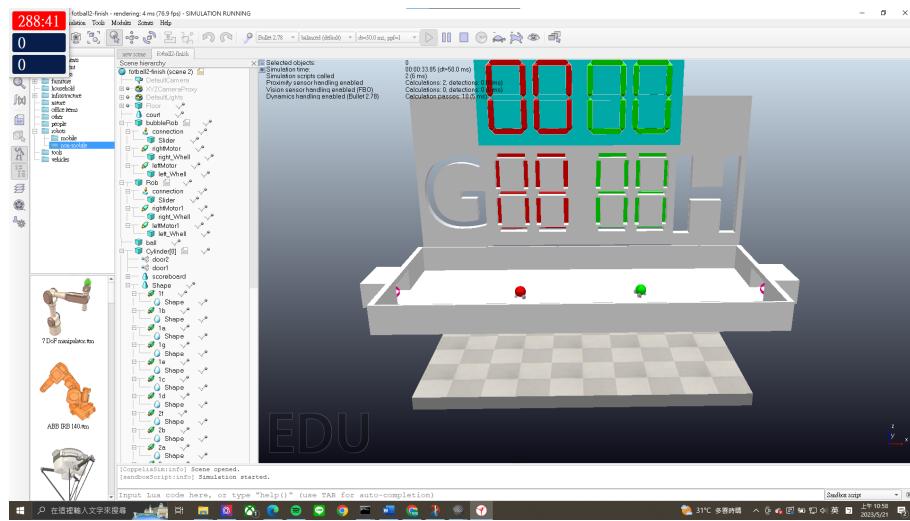


圖. 5.1: 完成作業