

國立虎尾科技大學

機械設計工程系

cd2023 2a3-pj3ag2 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：江芷柔 (41023103)

 李凱新 (41023106)

 王翔楷 (41023113)

 吳勁毅 (41023116)

 李學淵 (41023125)

 林秉賢 (41023132)

 張育銓 (41023151)

 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 與八位同組組員協同控制 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。

關鍵字：類神經網路、強化學習、CoppeliaSim、OpenAI Gym

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, it is imported into the CoppeliaSim simulation environment and configured accordingly. We use the zmqRemoteAPI to collaboratively control the bubbleRob with eight teammates in the same group. We engage in a football competition within the scene we have created and simultaneously incorporate a scoreboard to display the current score on the field.

Keyword: nerual network 、 reinforcement learning 、 CoppeliaSim 、 OpenAI
Gym

目 錄

摘要.....	i
Abstract	ii
第一章 前言	1
1.1 規則.....	1
第二章 場景建立	2
2.1 前言.....	2
2.2 建立球員.....	2
2.3 建立記分板.....	3
2.4 建立球場.....	4
2.5 整合場景.....	5
第三章 程式講解	6
3.1 設定句柄變數	6
3.2 回復記分板顏色	8
3.3 產生隨機座標	8
3.4 設定記分板顏色	8
3.5 分解剩餘時間數字	8
3.6 感測得分並修改記分板	9
3.7 將即時分數顯示在計分板	10
3.8 時間到則停止遊戲	10

3.9 回復記分板顏色	11
第四章 2a3-pj3ag2 製作心得	12
4.1 江芷柔心得.....	12
4.2 李凱新心得.....	12
4.3 王翔楷心得.....	12
4.4 吳勁毅心得.....	12
4.5 李學淵心得.....	12
4.6 林秉賢心得.....	12
4.7 張育銓心得.....	13
4.8 張昱棠心得.....	13
第五章 完成作業	14

圖 目 錄

圖 2.1 球員建立	2
圖 2.2 球員建立 2	3
圖 2.3 記分板建立	3
圖 2.4 匯入記分板	3
圖 2.5 球場繪製	4
圖 2.6 球門繪製	4
圖 2.7 建立球場	4
圖 2.8 場景建立完成	5
圖 3.1 設定句柄變數	6
圖 3.2 設定句柄變數註解	7
圖 3.3 回復記分板顏色	8
圖 3.4 產生隨機座標	8
圖 3.5 設定記分板顏色	8
圖 3.6 分解剩餘時間數字	9
圖 3.7 感測得分並修改記分板	9
圖 3.8 將即時分數顯示在計分板	10
圖 3.9 時間到則停止遊戲	11
圖 3.10 回復記分板顏色	11

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 前言

因為這次老師重新規劃了球場及球員的大小、重量…… 等等，我們重新規劃了整個球場及新建了球員模型，沒有沿用過去的設計。

2.2 建立球員

我們使用 Onshape 重新繪製了球員模型，這次規劃想改變過去 bubbleRob 看起來較呆版、圓潤的設計，因此設計了一台跑車作為我們的球員。File-Import-Mesh，選擇要匯入的檔案匯入球場，如(圖.2.1)。接著加入 joint

加入 joint: 滑鼠右鍵-Add-Joint-Revolute

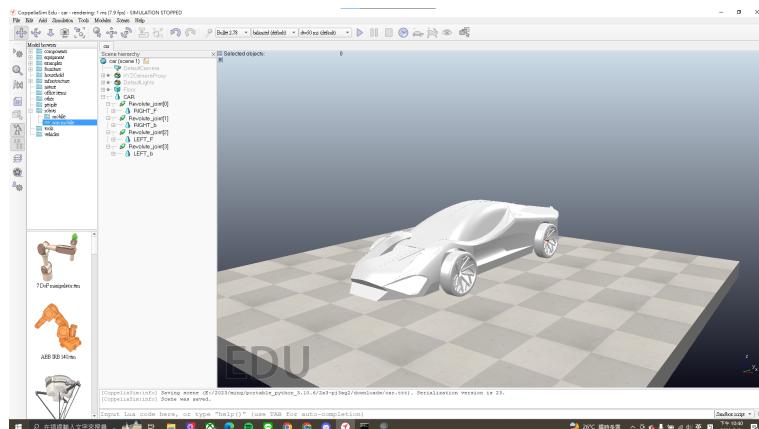


圖. 2.1: 球員建立

後發現比例錯誤，因此直接在 CoppeliaSim 內進行縮小，但因本來設計的跑車高度太低太扁平，可能會有無法推到球的狀況發生，因此不是使用等比縮小，而是直接將各部分拉至規定尺寸。如(圖.2.2)
比例放大縮小步驟: 點選物件左邊圖示-View modify geornrtry-若勾選 Keep proportions 為進行等比放大，若取消勾選擇可以個別調整大小。

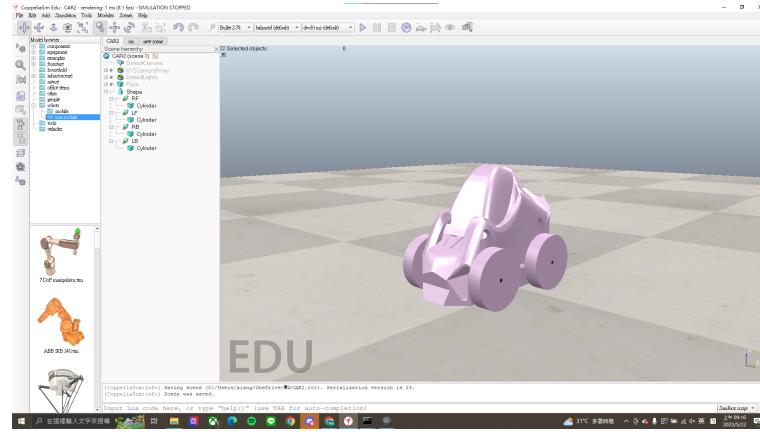


圖. 2.2: 球員建立 2

2.3 建立記分板

我們使用 Onshape 重新繪製了機械式記分板，如(圖.2.3)，接著匯入到CoppeliaSim 內進行爆炸拆件，拆件後加入 joint 如(圖.2.4)。

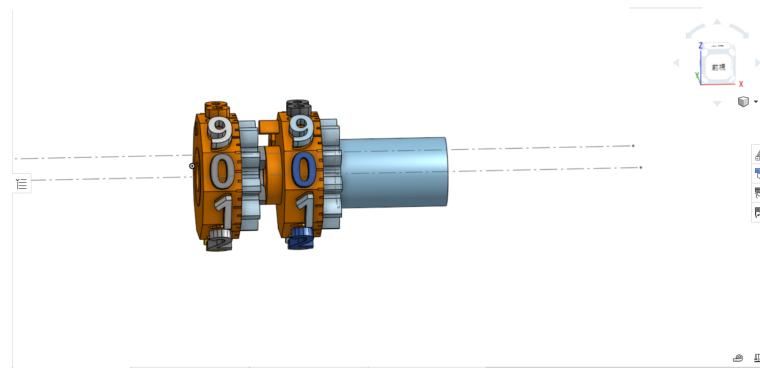


圖. 2.3: 記分板建立

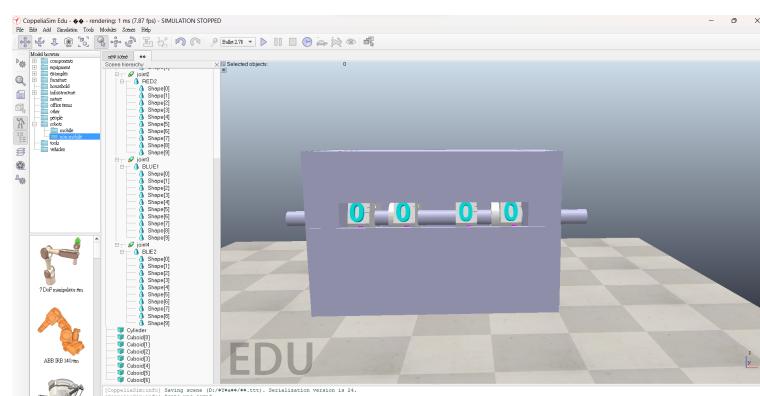


圖. 2.4: 匯入記分板

2.4 建立球場

我們使用 solidworks 繪製了球場底板及球門，如 (圖.2.5)、(圖.2.6)，匯入 CoppeliaSim 後接著建立感測器，如 (圖.2.7)。

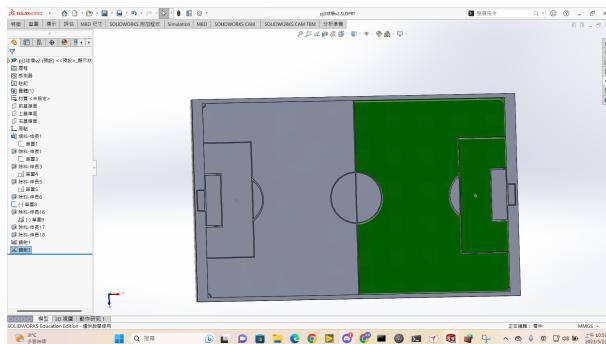


圖. 2.5: 球場繪製

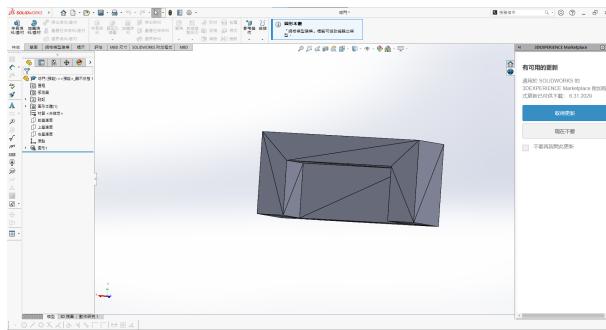


圖. 2.6: 球門繪製

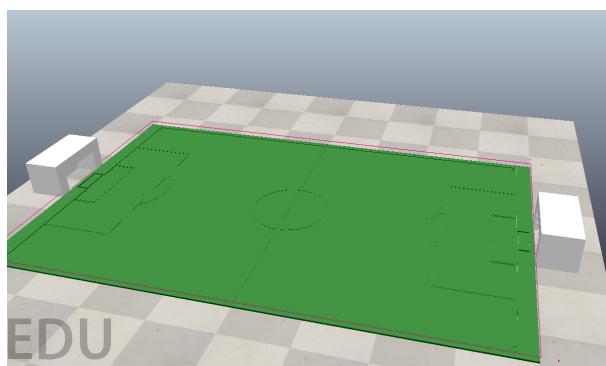


圖. 2.7: 建立球場

2.5 整合場景

接著將所有檔案拉到同一視窗內，球員進行分色，場地位置調整及加入感測器。

球員變色: 點選本體旁邊圖示-Adjust color-Amibient/diffuse component-拉動 RGB 調整顏色即可。

加入感測器:Add Proximity sensor-Ray type

成果如(圖.2.8)

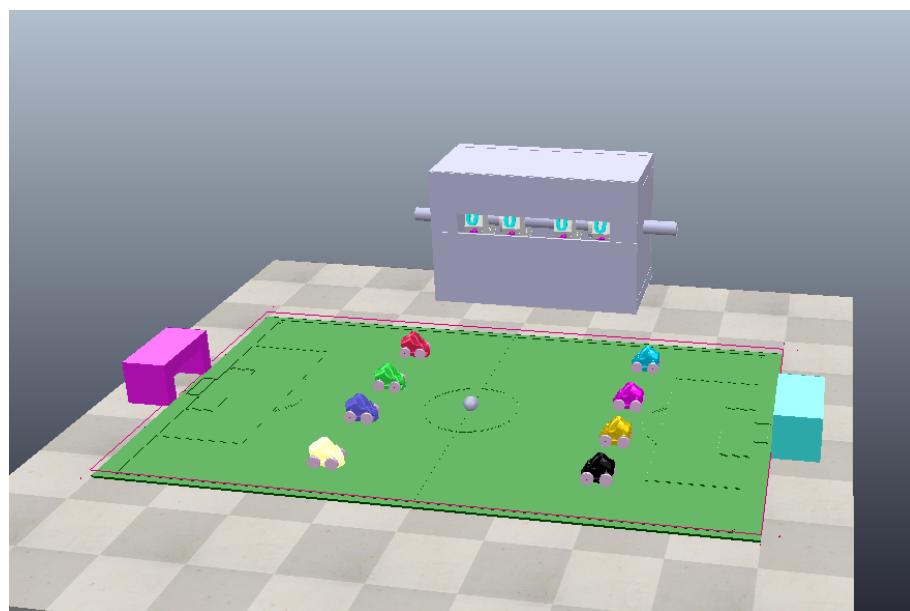


圖. 2.8: 場景建立完成

第三章 程式講解

3.1 設定句柄變數

```
function sysCall_init()
    count = 300
    score1 = 0
    score2 = 0
    timer = 0
    deltaTime = sim.getSimulationTimeStep()
    print(deltaTime)
    ball = sim.getObject('/Capsule')
    for i = 1, 4 do
        joint = sim.getObject('/box/joint' .. i)
        joints[i] = joint
    end
    for i = 0, 5 do
        sensor = sim.getObject('/Proximity_sensor[' .. i .. ']')
        sensors[i+1] = sensor
    end
    for i = 0, 1 do
        socoresensor = sim.getObject('/Shape[0]/Proximity_sensor[' .. i .. ']')
        socoresensors[i+1] = socoresensor
    end
    for i = 1,8 do
        BubbleRob = sim.getObject('/Shape[' .. i+2 .. ']')
        BubbleRobs[i] = BubbleRob
        initPosition[i] = sim.getObjectPosition(BubbleRob, -1)
        initOrientation[i] = sim.getObjectOrientation(BubbleRob, -1)
    end
    for i = 1, 7 do
        for x = 1, 4 do
            ii = tostring(x) .. serialNumber[i]
            local handle = sim.getObjectHandle("/scoreboard/".. ii )
            handles[x][i] = handle
        end
    end
    for i = 1, 7 do
        for x = 1, 4 do
            ii = tostring(x) .. serialNumber[i]
            local timehandle = sim.getObjectHandle("/time/".. ii )
            timehandles[x][i] = timehandle
        end
    end
    for x = 0, 1 do
        local dot = sim.getObjectHandle("/dot[" .. x .. "]")
        dots[x+1] = dot
    end
end
```

圖. 3.1: 設定句柄變數

這段程式碼，如(圖.3.1)，功能性及註解，如(圖.3.2)

```

function sysCall_init()
count = 300 -- 設置計數器初始值為300
score1 = 0 -- 設置第一個得分變量初始值為0
score2 = 0 -- 設置第二個得分變量初始值為0
timer = 0 -- 設置計時器初始值為0
deltaTime = sim.getSimulationTimeStep() -- 獲取模擬時間步長
print(deltaTime) -- 輸出時間步長
ball = sim.getObject('/Capsule') -- 獲取球體物體的句柄

```

```

-- 設取關節句柄
joints = {} -- 用於存儲所有關節句柄的表
for i = 1, 4 do
    joint = sim.getObject('/box/joint' .. i)
    joints[i] = joint
end

-- 設取接近傳感器句柄
sensors = {} -- 用於存儲所有接近傳感器句柄的表
for i = 0, 5 do
    sensor = sim.getObject('/Proximity_sensor[' .. i .. ']')
    sensors[i+1] = sensor
end

-- 設得分傳感器句柄
socoresensors = {} -- 用於存儲所有得分傳感器句柄的表
for i = 0, 1 do
    socoresensor = sim.getObject('/Shape[0]/Proximity_sensor[' .. i .. ']')
    socoresensors[i+1] = socoresensor
end

-- 設取 BubbleRob 物體句柄、初始位置和初始方向
BubbleRobs = {} -- 用於存儲所有 BubbleRob 物體句柄的表
initPosition = {} -- 用於存儲所有 BubbleRob 初始位置的表
initOrientation = {} -- 用於存儲所有 BubbleRob 初始方向的表
for i = 1, 8 do
    BubbleRob = sim.getObject('/Shape[' .. i .. ']')
    BubbleRobs[i] = BubbleRob
    initPosition[i] = sim.getObjectPosition(BubbleRob, -1)
    initOrientation[i] = sim.getObjectOrientation(BubbleRob, -1)
end

-- 設取計分板句柄
handles = {} -- 用於存儲所有計分板句柄的表
serialNumber = {1,2,3,4,5,6,7} -- 用於生成計分板句柄的序列號
for i = 1, 7 do
    handles[i] = {} -- 用於存儲單個 BubbleRob 的計分板句柄的表
    for x = 1, 4 do
        ii = tostring(x) .. serialNumber[i]
        local handle = sim.getObjectHandle("/scoreboard/" .. ii)
        handles[x][i] = handle
    end
end

```

圖 . 3.2: 設定句柄變數註解

3.2 回復記分板顏色

```
for i = 1, 7 do
    for x = 1, 4 do
        handle = handles[x][i]
        sim.setShapeColor(handle, nil, sim.colorcomponent_ambient_diffuse, colors[3])
        timehandle = timehandles[x][i]
        sim.setShapeColor(timehandle, nil, sim.colorcomponent_ambient_diffuse, colors[3])
    end
end
```

圖. 3.3: 回復記分板顏色

這段程式碼，如 (圖.3.10)，四、六行註解為設置計分板形狀的顏色為 color3。

3.3 產生隨機座標

```
function randomNumber()
    math.randomseed(os.time())
    return {tonumber(math.random(-1, 1) .. '.' .. math.random(0,9)),tonumber(0 .. '.' .. math.random(0,9)),0.2}
end
```

圖. 3.4: 產生隨機座標

這段程式碼，如 (圖.3.4)，第二行註解為使用當前時間作為隨機亂數。

3.4 設定記分板顏色

```
function dotcolor(color)
    for i = 1,2 do
        dot = dots[i]
        sim.setShapeColor(dot, nil, sim.colorcomponent_ambient_diffuse, colors[color])
    end
end
```

圖. 3.5: 設定記分板顏色

這個函式，如 (圖.3.5) 用於設置兩個點的顏色。它接受一個參數 color，用於指定要設置的顏色。在函式內部，使用 sim.setShapeColor 函式將顏色應用到每個點的形狀上，使用 colors[color] 來獲取指定的顏色。

3.5 分解剩餘時間數字

這個函式，如 (圖.3.6) 用於將數字轉換為計分板的十位數和個位數。它接受一個參數 number，代表要轉換的數字。首先，將數字轉換為

```

function scoreboard(number)
    local numberString = tostring(number)
    if #numberString < 2 then
        numberString = '0' .. numberString
    end
    local tensDigit = tonumber(numberString:sub(1, 1))
    local onesDigit = tonumber(numberString:sub(2, 2))
    return{tensDigit,onesDigit}

```

圖. 3.6: 分解剩餘時間數字

字符串 numberString。如果 numberString 的長度小於 2，則在前面添加一個'0'，確保有兩位數字。然後，使用 tonumber 和 string.sub 函式提取十位數和個位數，分別存儲在 tensDigit 和 onesDigit 中。最後，返回一個包含十位數和個位數的數組。

3.6 感測得分並修改記分板

```

if(scoreresult1>0) then
    regress(0)
    score2 = score2+1
end
if(scoreresult2>0)then
    regress(2)
    score1 = score1+1
end

```

圖. 3.7: 感測得分並修改記分板

這個函式，如 (圖.3.7) 包含兩個條件語句。第一個條件語句檢查 scoreresult1 是否大於 0，如果是，則執行 regress(0) 函式並將 score2 加一。第二個條件語句檢查 scoreresult2 是否大於 0，如果是，則執行 regress2) 函式並將 score1 加一。

3.7 將即時分數顯示在計分板

```
pink = scoreboard(score1)
blue = scoreboard(score2)
Toclear()
for i = 1, 2 do
    Number(i,pink[i],1)
    --Number(i,0)
    Number(i+2,blue[i],2)
end
```

圖. 3.8: 將即時分數顯示在計分板

這個函式，如 (圖.3.8) 首先將 score1 轉換為十位數和個位數，並將結果存儲在 pink 中。然後將 score2 轉換為十位數和個位數，並將結果存儲在 blue 中。接下來調用 Toclear 函式。

接下來的迴圈對於 i 等於 1 到 2，進行以下操作：

使用 Number i, pink i , 1 將 pink 中的十位數或個位數設置到數字顯示器中，並設置為顏色 1。

使用 Number i 加 2, blue i, 2 將 blue 中的十位數或個位數設置到數字顯示器中，並設置為顏色 2。

3.8 時間到則停止遊戲

這個函式，如 (圖.3.9) 包含一個條件語句。首先檢查 count 是否大於 0，如果是，則執行以下操作：

檢查 timer 是否大於等於 1，如果是，則將 timer 重置為 0，並將 count 減一。

根據 count 計算出分鐘數和秒數，並使用 string.format 函式將其格式化為分鐘: 秒的字符串形式。

將秒數和分鐘數分別轉換為十位數和個位數，並將結果分別存儲在 timesseconds 和 timesminutes 中。將 dottime 加一。

```

if count > 0 then
    if timer >= 1 then
        timer = 0
        count = count - 1
        local minutes = math.floor(count / 60)
        local seconds = count % 60
        local timeStr = string.format("%d:%02d", minutes, seconds)
        timesseconds = scoreboard(seconds)
        timesminutes = scoreboard(minutes)
        dottime = dottime+1
    end
    if dottime >= 2 then
        dottime = 0
    end
else
    sim.stopSimulation()
end

```

圖 . 3.9: 時間到則停止遊戲

接下來，檢查 dottime 是否大於等於 2，如果是，則將 dottime 重置為 0。如果 count 小於等於 0，則執行 sim.stopSimulation 函式停止仿真。

3.9 回復記分板顏色

```

for i = 1, 7 do
    for x = 1, 4 do
        handle = handles[x][i]
        sim.setShapeColor(handle, nil, sim.colorcomponent_ambient_diffuse, colors[3])
        timehandle = timehandles[x][i]
        sim.setShapeColor(timehandle, nil, sim.colorcomponent_ambient_diffuse, colors[3])
    end
end

```

圖 . 3.10: 回復記分板顏色

這個函式，如 (圖 .3.10) 使用兩個嵌套的迴圈。外層迴圈遍歷 i 從 1 到 7 的值，內層迴圈遍歷 x 從 1 到 4 的值。

對於每一個 i 和 x 的組合，執行以下操作：

從 handles 中獲取 handle 的值。使用 sim.setShapeColor 函式將 handle 的形狀顏色設置為 colors3。從 timehandles 中獲取 timehandle 的值。使用 sim.setShapeColor 函式將 timehandle 的形狀顏色設置為 colors3。換句話說，這段代碼將遍歷一個二維數組 handles 和 timehandles，並將相應的形狀顏色設置為 colors3。

第四章 2a3-pj3ag2 製作心得

4.1 江芷柔心得

4.2 李凱新心得

這次使用 SolidWorks 畫了一個球場，還使用 onshape 畫了旗子上面還有數字，再匯入 coppeliasim 。我感覺對於使用場景更得心應手了呢！

4.3 王翔楷心得

這次作業我擔任建立球員及球場感測器還有最後 Latex 報告撰寫的工作，設計跑車模樣的球員讓我的繪圖技術透過這次的經驗更加精進，球場感測器的建立已經透過前幾次的作業徹底摸清楚，建設上非常的流暢也省時，最後是 Latex 報告的編寫，每次寫報告都能更精進使用 latex 的技術，總之這門課使我獲益良多。

4.4 吳勁毅心得

這次 pj3 的分組因為有 8 個人所以比之前更輕鬆，畢竟要負責的部分已經更少了，但途中還是遇到一些問題，像是有時無法連到主機，導致無法對戰，還有一些程式小問題，但我們會盡力解決，讓這個作業做得更好

4.5 李學淵心得

這次 pj3 我學習到了 lua 在 coppeliasim 的各種用法，並整合了更多功能在程式之中，縮短程式的長度來簡潔美化。

4.6 林秉賢心得

這次的八人組合，我覺得做的東西有難度，但是如果要八個人一起做我覺得可以很快完成，我要很謝謝我的組員們很凱瑞我，沒有他們我覺得這個東西沒辦法處理得很好。

4.7 張育銓心得

這次的 pj3 作業很難，大部分都是靠同組的同學完成的，在連網路也發現到了一些問題，並解決了它，非常的艱難，也很感謝同組的同學。

4.8 張昱棠心得

這次的作業裡的機械式計分器挑戰對我的繪圖挑戰真的很大，來回改良了好幾版，匯入 coppliasim 模擬作動。對我來說都是很棒的跳戰，我學到了很多解決事情的方法。

第五章 完成作業

經過幾次的修改及優化，完成作業。

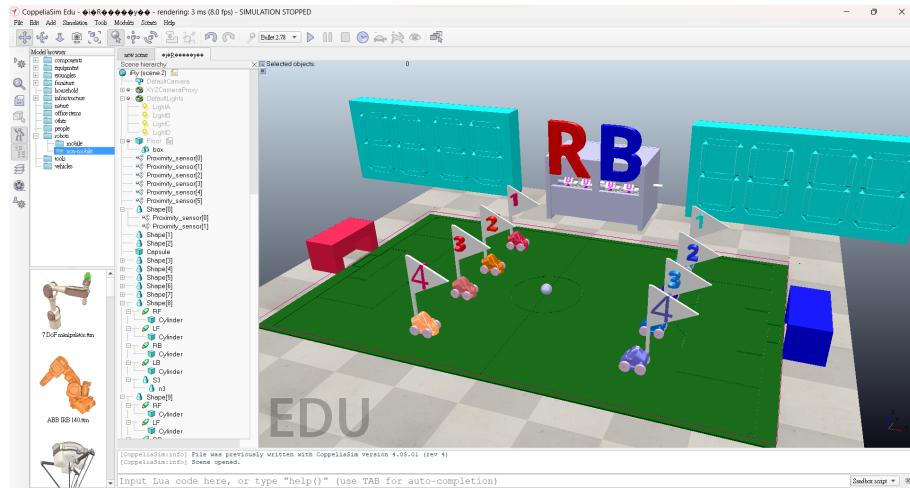


圖. 5.1: 完成作業