

國立虎尾科技大學
機械設計工程系
cd2023 2a-pj1ag14 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：張昱棠 (41023153)
王翔楷 (41023113)

中華民國 112 年 3 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。

此專案是運用 w3 作業所做的泡泡機器人，將其導入 CoppeliaSim 模擬環境並給予對應設置，將其機電系統簡化並導入 Lua 及 python 程式，再到 CoppeliaSim 模擬環境中進行測試演算法在實際運用上的可行性。並嘗試透過架設伺服器將 CoppeliaSim 影像串流到網頁供使用者觀看或操控。

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 involves taking the bubbleRob created for the W3 assignment and importing it into the CoppeliaSim simulation environment with corresponding settings, simplifying its electromechanical system, and incorporating Lua and Python programming for testing algorithms in the CoppeliaSim simulation environment to determine their feasibility in practical application. Additionally, the project aims to stream CoppeliaSim images to a webpage for users to view or manipulate by setting up a server.

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，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。再來是李學淵和李凱新同學，和我們共同討論遇上問題並思考解決方案，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	ii
誌謝	iii
第一章 前言	1
1.1 規則說明	1
第二章 bubbleRob 機器人建立	2
2.1 將 Coppeliasim 版本更改為可執行 python 程式	2
2.2 本體建立	3
第三章 程式講解	6
3.1 程式講解	6
第四章 bubbleRob 製作心得	9
4.1 張昱棠心得	9
4.2 王翔楷心得	9
第五章 PJ1-Football Rob	10
5.1 球場建立	10
第六章 導入程式	13
6.1 bubbleRob 與感測器於球場內程式	13
6.2 本地程式	16
6.3 遠端程式	17

第七章 完成作業 20

圖 目 錄

圖 2.1 設定	2
圖 2.2 本體建立	3
圖 2.3 Body is respondable 、 Body is dynamic	4
圖 2.4 Collidable 、 Measurable 、 Detectable	5
圖 3.1 程式講解	6
圖 3.2 程式講解 2	7
圖 5.1 球場繪製	10
圖 5.2 導入球場	11
圖 5.3 加入感測器	11
圖 5.4 加入 bubbleRob	12
圖 5.5 加入球體	12
圖 6.1 偵測用於遠端操控的 bubbleRob 以及左右馬達的變值	13
圖 6.2 設定 sensor1 、 2 為 door1 、 2	13
圖 6.3 計時器時間及得分設定	13
圖 6.4 設定 UI 介面	14
圖 6.5 感測器偵測到物體 UI 介面加分及位置重製	14
圖 6.6 完成後 UI 介面展示	16
圖 6.7 導入 python 程式	16

圖 6.8 本地控制程式	17
圖 6.9 查看本地 IP	18
圖 6.10 IP config	18
圖 6.11 遠端控制程式	19
圖 7.1 完成作業	20

表 目 錄

第一章 前言

1.1 規則說明

Pong game 的遊戲規則簡單，透過 bubbleRob 將球打入對方球門即得一分，時間內其中一方分數高獲勝。

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 bubbleRob 機器人建立

2.1 將 Coppeliasim 版本更改為可執行 python 程式

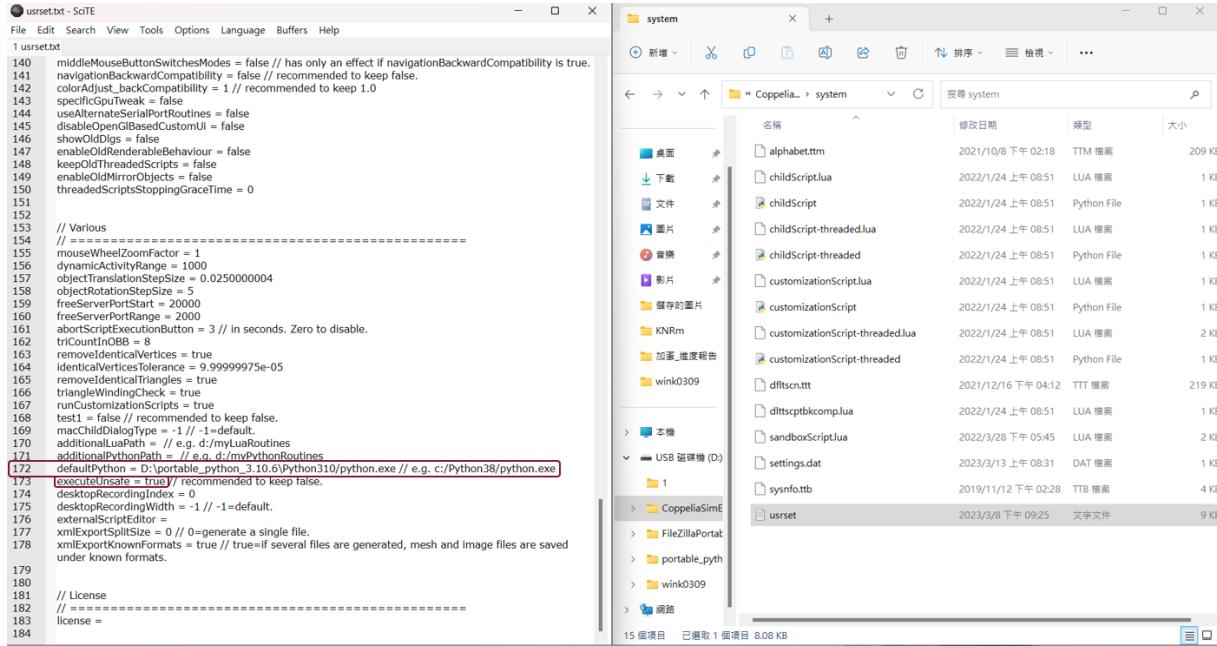


圖. 2.1: 設定

Coppeliasim 自 4.3 版之後除了 Lua scripting, 增加了 Python scripting 功能。但是必須在 usrset.txt 中設定 defaultPython 指向 Python.exe, pip install cbor zmq, 並且設定 executeUnsafe = true. 完成後開啟 Coppeliasim 4.3.0 rev12, 並利用 scenes/simplePythonExample.ttt 進行測試如 (圖.2.1)。

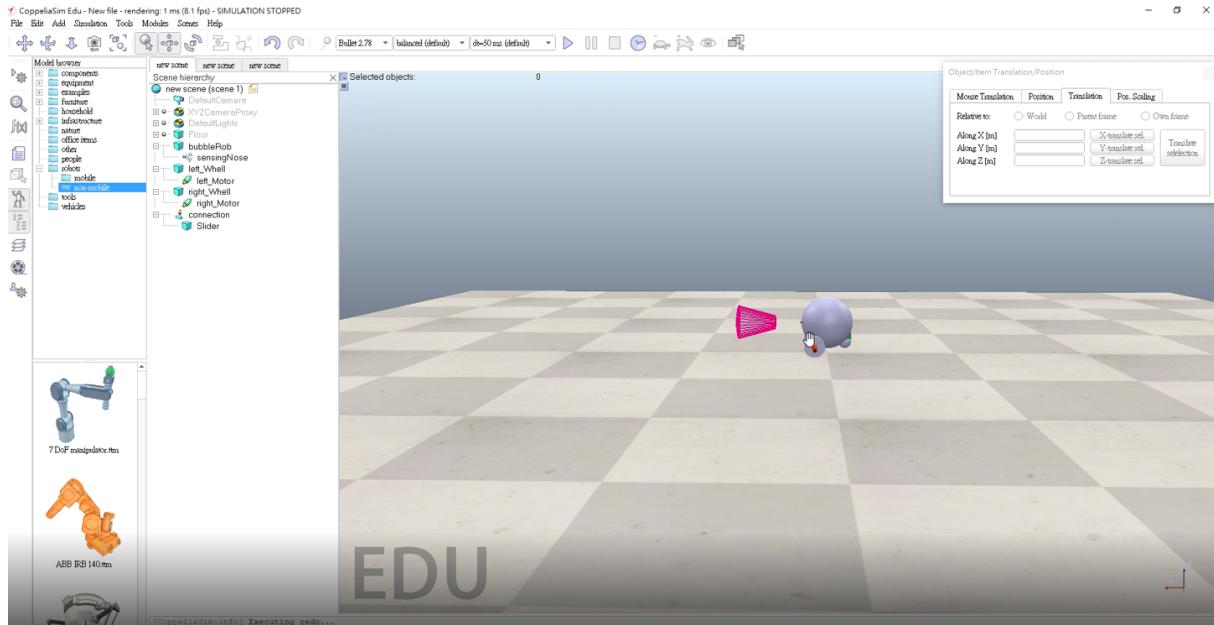


圖. 2.2: 本體建立

2.2 本體建立

本體如(圖.2.2)及感測器是進行專案的重要基礎，各項參數要設定得宜才能得到正確的結果。

以下介紹幾種較為常見的參數設定及其特性：

- Body is respondable (圖.2.3) :

respond 是回答，回應的意思。可響應形狀將導致與其他可響應形狀的碰撞反應。即 respondable 屬性用於控制物體的碰撞回應，nonrespondable 物體在與其它物體發生碰撞時，物理引擎不會進行計算，會出現穿透的現象。

- Body is dynamic (圖.2.3) 是具有動態屬性的物體會在重力影響下墜落或在其它外力、力矩作用下發生運動狀態的改變。而靜態物體則會在場景中靜止不動，如地面，或跟隨其父節點運動（或跟隨其父節點在場景層次結構中的移動）。

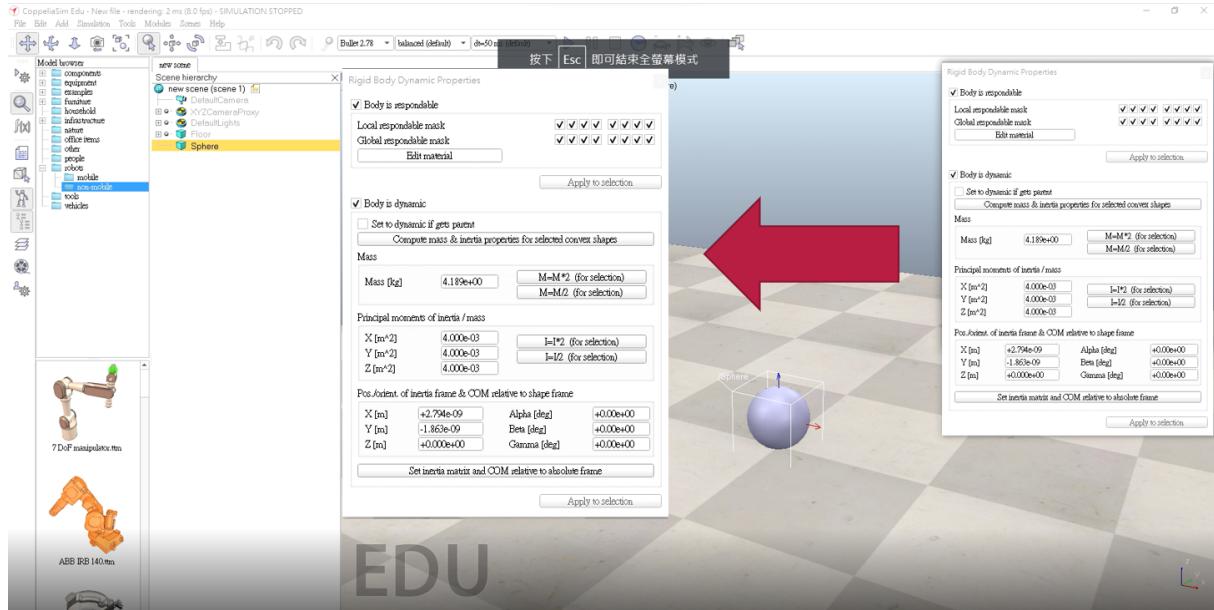


圖. 2.3: Body is respondable 、Body is dynamic

- Collidable(圖.2.4) :

碰撞檢測，可以檢測兩個碰撞體實體（Collidable objects are objects that can be tested for collision against other collidable objects）之間的碰撞，類似於 SolidWorks 等三維設計軟體中的干涉檢查。碰撞檢測只會檢測碰撞狀態，而不會直接對碰撞做出反應（The collision detection module will only detect collisions; it does however not directly react to them）。碰撞檢測模組中可以註冊碰撞物件，即 collidable entity-pairs (collider entity and collidee entity)。在模擬過程中，註冊的碰撞對象之間的碰撞狀態可以由不同的顏色可視化顯示，也可以通過 Graph 來進行記錄。

- Measurable(圖.2.4) :

通常指可以測量的物理量或仿真參數，例如機器人的位置、速度、加速度、力和角度等。這些可測量的量可以用來監測機器人的行為，評估控制算法的性能，或者用於自動化調整機器人的運動。

- Detectable(圖.2.4) :

指可以被檢測到的物理量或仿真參數，這些可檢測的量可以用來檢測機器人與周圍環境的交互，例如檢測是否有障礙物、檢測是否接近日標等。

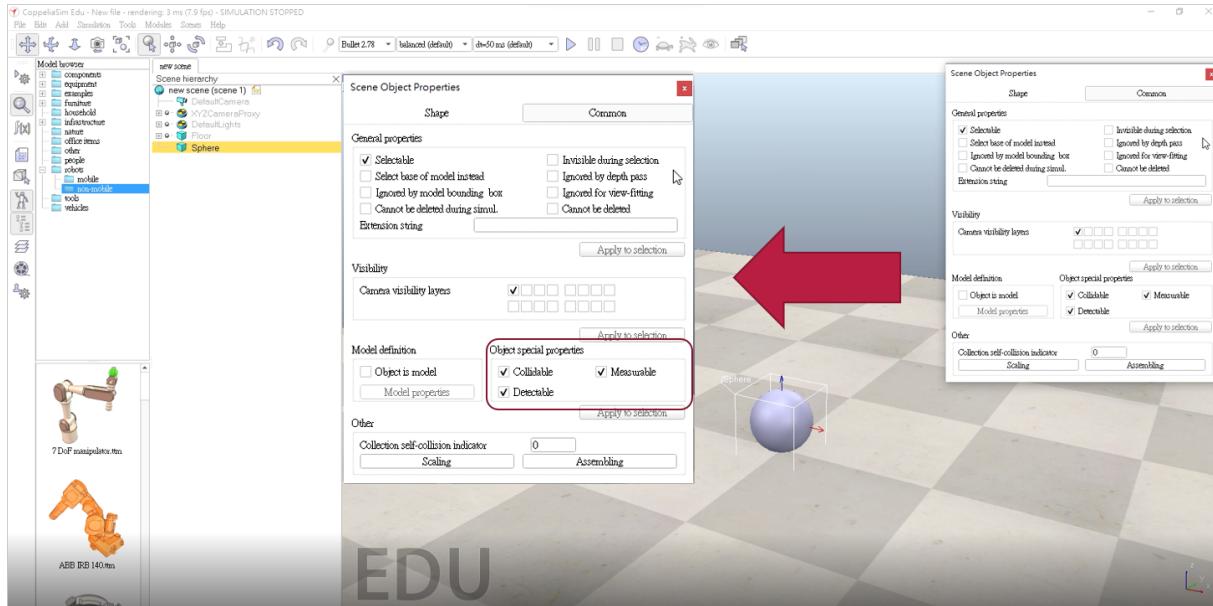


圖 . 2.4: Collidable 、 Measurable 、 Detectable

第三章 程式講解

3.1 程式講解

```
1 function sysCall_vision(inData)
2     simVision.sensorImgToWorkImg(inData.handle) -- copy the vision sensor image to the work image
3     simVision.edgeDetectionOnWorkImg(inData.handle,0.2) -- perform edge detection on the work image
4     simVision.workImgToSensorImg(inData.handle) -- copy the work image to the vision sensor image buffer
5 end
6
7 function sysCall_init()
8 end
```

圖. 3.1: 程式講解

首先這是一個 Lua 語言，在 CoppeliaSim 仿真環境中運行。該腳本定義了兩個函數：sysCallinit 和 sysCallvision。

sysCallinit 函數是仿真環境初始化時自動調用的函數，該函數目前是空的，即不執行任何操作。

sysCallvision 函數是 CoppeliaSim 的視覺模組塊（VisionModule）在每次運行時會調用的函數，該函數的作用是對視覺傳感器的圖像進行邊緣檢測（edge detection）。

具體來說，函數中的三個函數調用分別為：

- simVision.sensorImgToWorkImg (圖.3.1) :

將視覺傳感器的圖像複製到工作圖像中。

- simVision.edgeDetectionOnWorkImg (圖.3.1) :

對工作圖像進行邊緣檢測，檢測閾值為 0.2。

- simVision.edgeDetectionOnWorkImg (圖.3.1) :

將處理後的工作圖像複製回視覺傳感器的圖像緩衝區中。

其中，inData 是一個包含了視覺模組塊的一些信息的 table 對象，例如 handle（視覺傳感器的句柄）等。simVision 是 CoppeliaSim 中提供的用於處理視覺相關任務的庫。

```

1 function speedChange_callback(ui,id,newVal)
2     speed=minMaxSpeed[1]+(minMaxSpeed[2]-minMaxSpeed[1])*newVal/100
3 end
4
5 function sysCall_init()
6     -- This is executed exactly once, the first time this script is executed
7     bubbleRobBase=sim.getObject('.')
8     leftMotor=sim.getObject("./leftMotor") -- Handle of the left motor
9     rightMotor=sim.getObject("./rightMotor") -- Handle of the right motor
10    noseSensor=sim.getObject("./sensingNose") -- Handle of the proximity sensor
11    minMaxSpeed={50*math.pi/180,300*math.pi/180} -- Min and max speeds for each motor
12    backUntilTime=-1 -- Tell whether bubbleRob is in forward or backward mode
13    robotCollection=sim.createCollection(0)
14    sim.addItemToCollection(robotCollection,sim.handle_tree,bubbleRobBase,0)
15    distanceSegment=sim.addDrawingObject(sim.drawing_lines,4,0,-1,1,{0,1,0})
16    robotTrace=sim.addDrawingObject(sim.drawing_linestrip:sim.drawing_cyclic,2,0,-1,200,{1,1,0})
17    graph=sim.getObject('./graph')
18    distStream=sim.addGraphStream(graph,'bubbleRob clearance','m',0,{1,0,0})
19    -- Create the custom UI:
20    xml = '<ui title="" sim.getObjectAlias(bubbleRobBase,1)..> speed" closeable="false" resizable="false" activate="false">
21    <hslider minimum="0" maximum="100" onchange="speedChange_callback" id="1"/>
22    <label text="" style="margin-left: 300px;"/>
23    </ui>
24    ]]
25    ui=simUI.create(xml)
26    speed=(minMaxSpeed[1]+minMaxSpeed[2])*0.5
27    simUI.setSliderValue(ui,1,100*(speed-minMaxSpeed[1])/(minMaxSpeed[2]-minMaxSpeed[1]))
28 end
29
30 function sysCall_sensing()
31     local result,distData=sim.checkDistance(robotCollection,sim.handle_all)
32     if result>0 then
33         sim.addDrawingObjectItem(distanceSegment,nil)
34         sim.addDrawingObjectItem(distanceSegment,distData)
35         sim.setGraphStreamValue(graph,distStream,distData[7])
36     end
37     local p=sim.getObjectPosition(bubbleRobBase,-1)
38     sim.addDrawingObjectItem(robotTrace,p)
39 end
40
41 function sysCall_actuation()
42     result=sim.readProximitySensor(noseSensor) -- Read the proximity sensor
43     -- If we detected something, we set the backward mode:
44     if (result>0) then backUntilTime=sim.getSimulationTime()+4 end
45
46     if (backUntilTime<sim.getSimulationTime()) then
47         -- When in forward mode, we simply move forward at the desired speed
48         sim.setJointTargetVelocity(leftMotor,speed)
49         sim.setJointTargetVelocity(rightMotor,speed)
50     else
51         -- When in backward mode, we simply backup in a curve at reduced speed
52         sim.setJointTargetVelocity(leftMotor,-speed/2)
53         sim.setJointTargetVelocity(rightMotor,-speed/8)
54     end
55 end
56
57 function sysCall_cleanup()
58     simUI.destroy(ui)
59 end

```

圖. 3.2: 程式講解 2

如 (圖.3.2): 1 至 3 行這個函數會在使用者調整了速度控制 UI 元素後被呼叫。它接收 3 個參數：ui，這是 UI 元素的句柄；id，這是 UI 元素的 ID；newVal，這是 UI 元素的新值。函數會根據新值計算出一個速度值 speed。

第 5 行這個函數是模擬程式的初始化函數，它只會被執行一次，當模擬程式啟動時。這個函數主要是執行一些初始化設置，例如設置模型的基礎物體、感測器和控制器，以及創建一些繪圖對象和 UI 元素。

第 7 行這行代碼會獲取模型的根物體的句柄。在這個例子中，模型的根物體是代表機器人的物體。

第 8、9、10 行這些行會獲取左右馬達和前方感測器的句柄。這些句柄會在後續的代碼中用於控制馬達和讀取感測器。

第 11 行這行會定義最小和最大速度。在這個例子中，速度是以弧度每秒為單位表示的，最小速度為 50 度每秒，最大速度為 300 度每秒。

第 12 行這行代碼定義了一個變數 backUntilTime，用於在後續的代碼中區分機器人是向前移動還是向後移動。

第 13、14 行這些行代碼用於創建一個物體集合並將機器人物體添加到其中。這將使得感測器能夠檢測到機器人周圍的其他物體

第 41 行這個函數控制 BubbleRob 的行動。它首先讀取接近傳感器的數據，以檢測是否有東西在 BubbleRob 的前方。如果傳感器檢測到障礙物，則將 backUntilTime 設置為當前仿真時間加上 4 秒，表示 BubbleRob 現在處於倒車模式。否則，如果 backUntilTime 小於當前仿真時間，則 BubbleRob 處於前進模式，並且將左右馬達的目標速度設置為 speed。否則，BubbleRob 處於倒車模式，並且左右馬達的目標速度設置為一定的值，以實現向後行駛的效果。

第 57 行這個函數在仿真結束時被調用，以清理代碼中使用的任何資源。在這個例子中，它摧毀了 UI 對象，以避免內存洩漏。

第四章 bubbleRob 製作心得

4.1 張昱棠心得

我們在製作過程中遇到了相當多的問題，光語言的部分我們就開了一個翻譯的網頁在旁邊一起配著原文版看，在一開始時，有點不太理解調整數值是必須輸入註解內的數值還是本文內的，導致數值有些輸入錯誤，在一連串的錯誤後，我們果斷選擇，直接開一個新檔案重做，由於當天我們留在學校做，大概從七點開始一路錯誤重來錯誤重來到快凌晨一點才回家，終於在隔天成功做出結果來了，在學習 coppeliasim 的部分，接續上學期學到的在這學期應用，並且學習了許多新的知識以及應用。

4.2 王翔楷心得

製作初期其實並不順利，我們這組和另外兩個同學一起從七點弄到凌晨一點才回家，過程中最難克服的是對軟體的不熟悉，隔天利用下午沒課的時間慢慢摸索，最後才順利完成建置，再利用晚上上課的時間向老師請教感測器內部程式的問題後回到座位上研究，最後才順利完成功課。

第五章 PJ1-Football Rob

5.1 球場建立

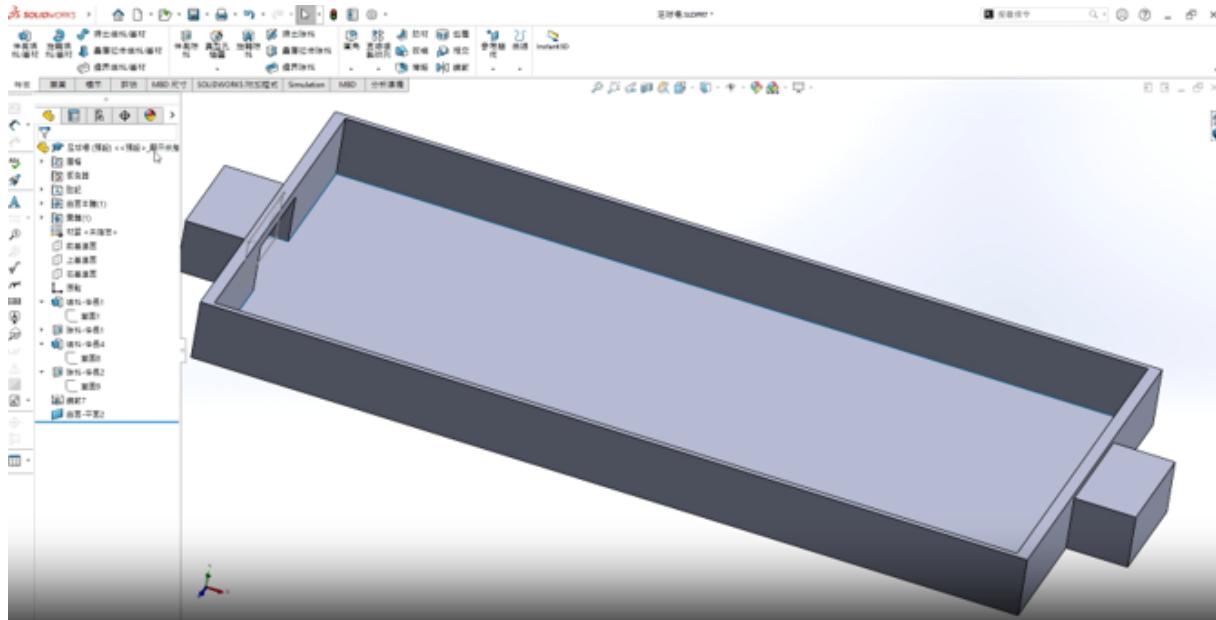


圖. 5.1: 球場繪製

如 (圖.5.1) 使用 SolidWorks 繪製我們足球場。接著將繪製完成將球場 STL 檔匯入 CoppeliaSim 並調整位置，但需將球場的 Collidable、Measurable 開啟不過 Detectable 需關閉，不然球門的感測器會偵測到球場本體。如 (圖.5.2)

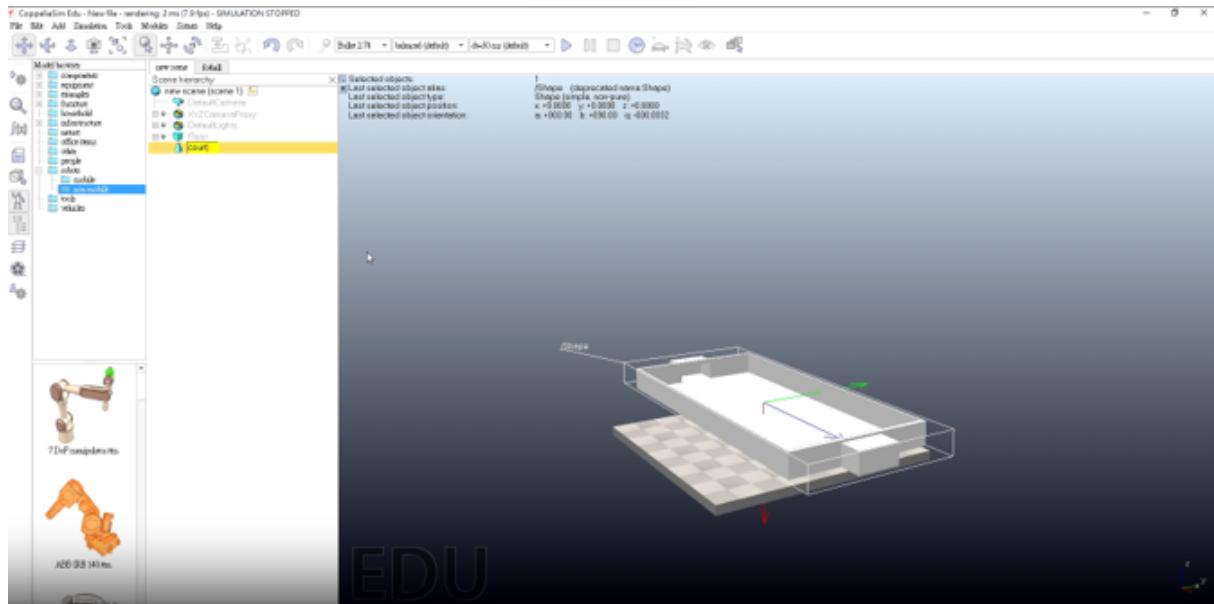


圖. 5.2: 導入球場

接著依序加入感測器(圖.5.3)、bubbleRob(圖.5.4)、足球本體(圖.5.5)
並且定義位置。

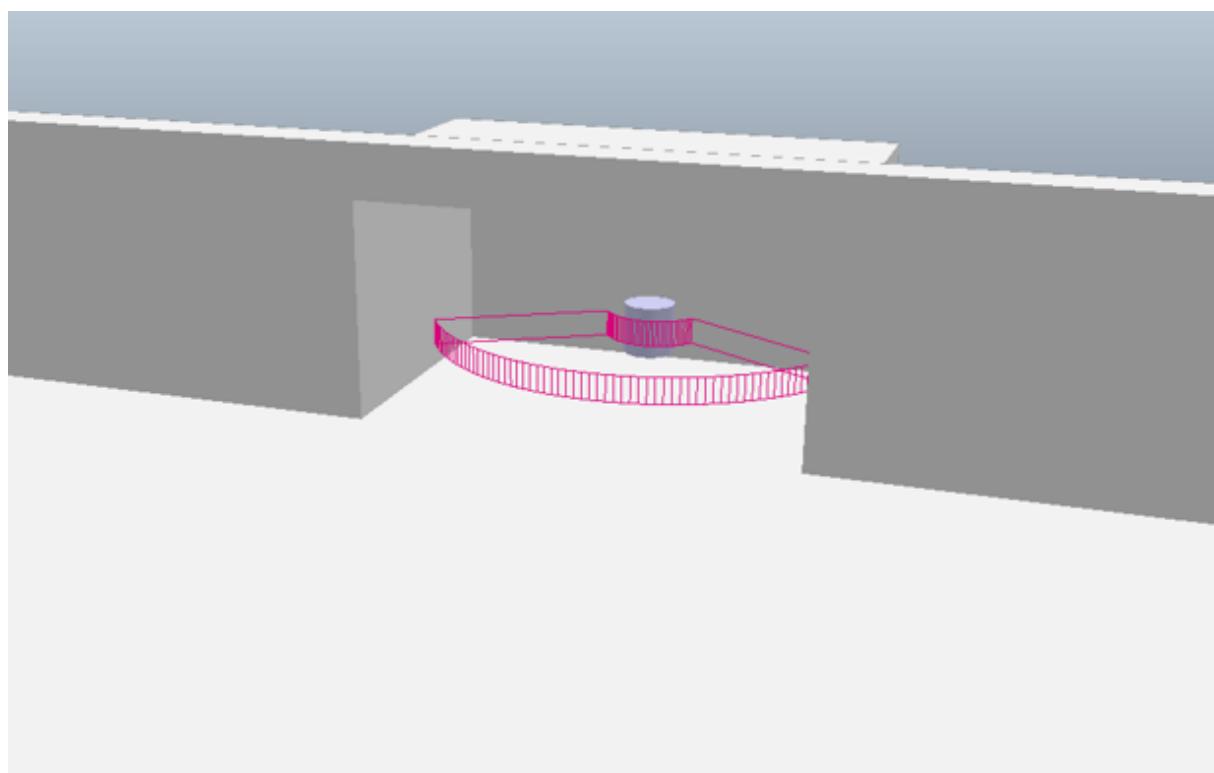


圖. 5.3: 加入感測器

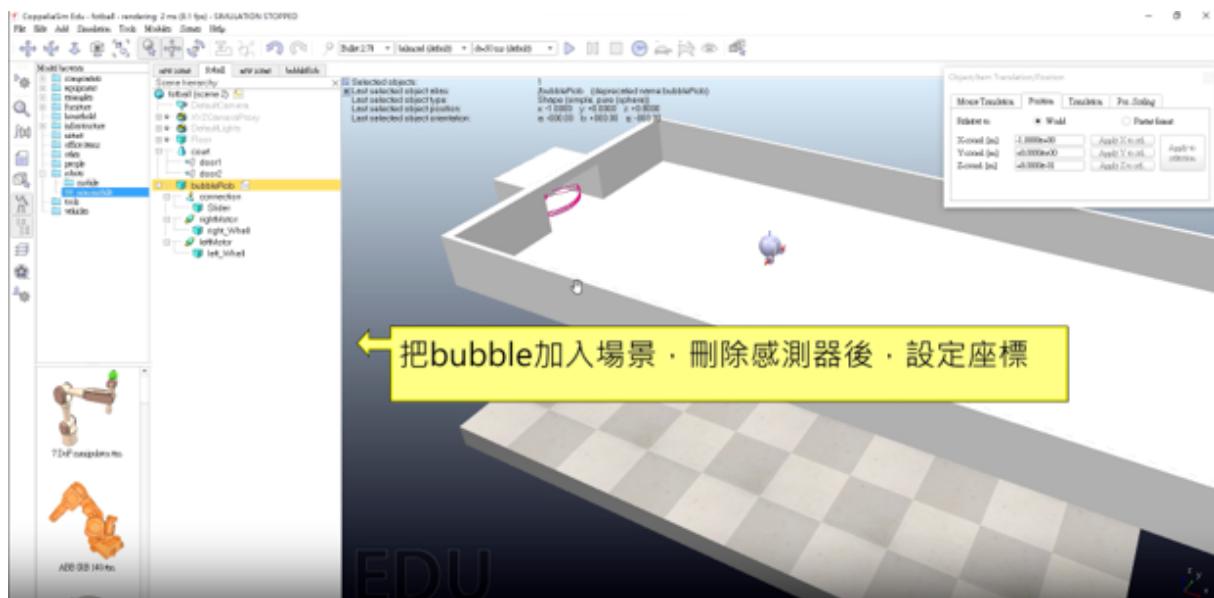


圖. 5.4: 加入 bubbleRob

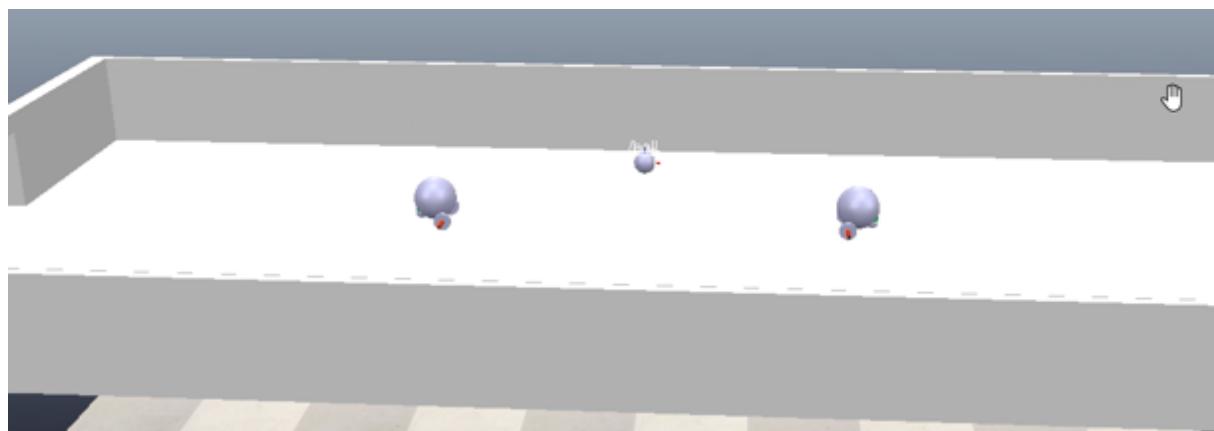


圖. 5.5: 加入球體

第六章 導入程式

6.1 bubbleRob 與感測器於球場內程式

```
function sysCall_init()
    print(sim.getObject('./rightMotor'))
    print(sim.getObject('./leftMotor'))
    print(sim.getObject('.'))
end
```

圖. 6.1: 偵測用於遠端操控的 bubbleRob 以及左右馬達的變值

```
sensor1 = sim.getObject('./door1')
sensor2 = sim.getObject('./door2')
bubbleRobBase = 16
ball = 30
initialBubbleRobPosition = sim.getObjectPosition(bubbleRobBase, -1)
initialBubbleRobOrientation = sim.getObjectOrientation(bubbleRobBase, -1)
initia = sim.getObjectPosition(23, -1)
initial = sim.getObjectOrientation(23, -1)
initialballPosition = sim.getObjectPosition(ball, -1)
initialballOrientation = sim.getObjectOrientation(ball, -1)
```

圖. 6.2: 設定 sensor1、2 為 door1、2

```
-- do some initialization here
count = 18000
score1 = 0
score2 = 0
```

圖. 6.3: 計時器時間及得分設定

```

xml = [
    <ui closeable="false" resizable="false" activate="false">
        <label text="0:00" style="" (background-color: #000; color: #FFF; font-size: 30px; font-weight: bold; padding: 0px; border-radius: 4px)" id="10"/>
        <label text="0" style="" (background-color: #000; color: #FFF; font-size: 30px; font-weight: bold; padding: 0px; border-radius: 4px)" id="20"/>
        <label text="0" style="" (background-color: #000; color: #FFF; font-size: 30px; font-weight: bold; padding: 0px; border-radius: 4px)" id="30"/>
    </ui>
]
ui = simUI.create(xml)
simUI.setPosition(ui, 0, 0, true)

```

圖. 6.4: 設定 UI 介面

```

function sysCall_actuation()
    result1=sim.readProximitySensor(sim.getObject('./door1'))
    result2=sim.readProximitySensor(sim.getObject('./door2'))
    -- 0 or 1
    if(result1>0)then
        sim.pauseSimulation()
        sim.setObjectPosition(bubbleRobBase, -1, initialBubbleRobPosition)
        sim.setObjectOrientation(bubbleRobBase, -1, initialBubbleRobOrientation)
        sim.setObjectPosition(ball, -1, initialballPosition)
        sim.setObjectOrientation(ball, -1, initialballOrientation)
        sim.setObjectPosition(23, -1, initia)
        sim.setObjectOrientation(23, -1, initial)
        score1 = score1 +1
    end
    if(result2>0)then
        sim.pauseSimulation()
        sim.setObjectPosition(bubbleRobBase, -1, initialBubbleRobPosition)
        sim.setObjectOrientation(bubbleRobBase, -1, initialBubbleRobOrientation)
        sim.setObjectPosition(ball, -1, initialballPosition)
        sim.setObjectOrientation(ball, -1, initialballOrientation)
        sim.setObjectPosition(23, -1, initia)
        sim.setObjectOrientation(23, -1, initial)
        score2 = score2 +1
    end
    if count > 0 then
        count = count - 1
        local minutes = math.floor(count / 60)
        local seconds = count % 60
        local timeStr = string.format("%d:%02d", minutes, seconds)
        simUI.setLabelText(ui, 10, timeStr)
        simUI.setLabelText(ui, 20, tostring(score1))
        simUI.setLabelText(ui, 30, tostring(score2))
    else
        sim.stopSimulation()
    end
end

```

圖. 6.5: 感測器偵測到物體 UI 介面加分及位置重製

如 (圖.6.1): 第 2 行這個語句呼叫了 sim.getObject 函數並傳入了一個字串參數'./rightMotor'。這個函數是用於獲取場景中指定名稱的物體也就是我們的右馬達。print 函數則是用於將獲取到的物體對象打印出來，以便進一步進行調試或是測試。第 3 行這個語句和第一個語句類似，只是傳入的參數不同，這個函數是用於獲取場景中另一個名稱為 leftMotor 的物體也就是左馬達。第 4 行這個語句呼叫了 sim.getObject 函

數並傳入了一個字串參數''。這個參數代表獲取當前場景中的根對象。這個函數用於獲取當前場景的根對象，以便進行對場景的操作或是進一步獲取子對象。函數 sysCallInit 用於初始化仿真環境和一些變量，而函數 sysCallActuation 用於執行機器人的運動和動作，以及更新計時器和得分標籤的顯示。

如(圖.6.2): sensor1 和 sensor2 變量是用來存儲兩個傳感器的句柄，這些傳感器可能被機器人用來檢測周圍環境。bubbleRobBase 變量存儲了機器人的基座句柄。ball 變量存儲了一個球的句柄。initialBubbleRobPosition 和 initialBubbleRobOrientation 分別是機器人基座的初始位置和方向。initialBallPosition 和 initialBallOrientation 分別是球的初始位置和方向。

如(圖.6.3): 這段代碼用於進行一些初始化操作。具體來說，我們將一個名為 count 的變量初始化為 18000，這個變量用於存儲倒計時的剩餘時間，單位為秒。我們還將 score1 和 score2 變量初始化為 0，這兩個變量用於表示兩個玩家的得分。

如(圖.6.4): 這段程式碼是用來建立一個使用者介面 UI，讓使用者可以看到遊戲時間和兩方的得分。程式中使用了 XML 格式的字串來定義介面的樣式，其中包括了三個標籤 Label，分別是時間、玩家 1 得分和玩家 2 得分並為每個標籤 Label 指定了一個唯一的識別碼 ID。程式中的... 語法是 Lua 的多行字串表示法，可以方便地在一段文字中包含多行內容。在這裡使用... 語法定義了一個名為 xml 的多行字串，這個字串中包含了 UI 元素的定義。接下來，程式呼叫 simUI.createXML 方法來創建一個使用者介面，ui 變數存儲了返回值，這個返回值是介面的識別碼 ID。接著，程式呼叫 simUI.setPosition ui, 0,0, true 方法來設定介面的位置，這裡將介面的位置設定為左上角，即 x 和 y 座標都為 0。

如(圖.6.5):接下來是sysCallactuation函數，這個函數會在每一幀被調用。這個函數會檢測兩個門的狀態，如果某個門被打開了，那麼就暫停遊戲，並重置機器人和球的位置，以及增加對應玩家的得分。另外，如果遊戲時間已經用完，那麼也會停止遊戲。在每一幀結束時，還會更新遊戲時間和得分在用戶界面上的顯示。最後，需要注意的是，這些代碼並不是獨立運行的，而是作為一段代碼塊被嵌入到了一個名為BubbleRob的場景中。因此，在運行這段代碼之前，需要先打開這個場景。

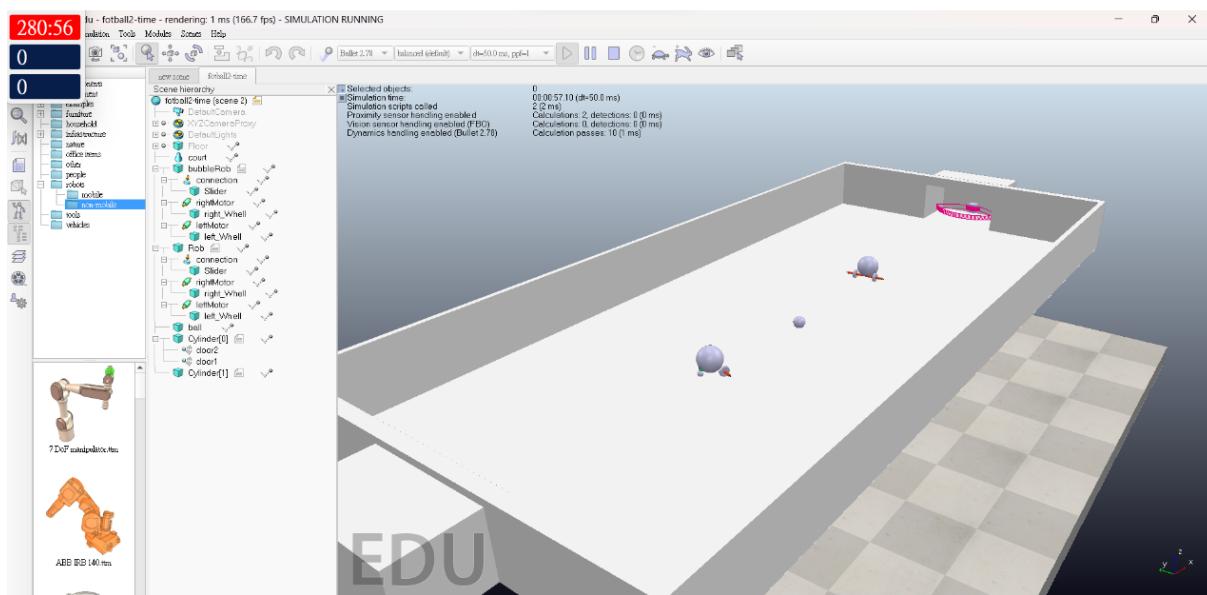


圖. 6.6: 完成後 UI 介面展示

6.2 本地程式

remoteApi.dll	2021/10/8 下午 02:17	應用程式擴充	76 KB
sim.py	2021/10/8 下午 02:17	Python File	74 KB
simConst.py	2019/11/12 下午 02:25	Python File	43 KB

圖. 6.7: 導入 python 程式

連機須將這三個檔案丟入 python310 內，如(圖.6.7)。

```
1 test.py * 2 newbing_bubbleRob_remoteapi1.py 3 1998.py
```

```
import sim
import sys
import time
import keyboard

# Connect to the simulation
sim.simxFinish(-1)
clientID = sim.simxStart('127.0.0.1', 19997, True, True, 5000, 5)

- if clientID != -1:
    print('Hello,bubbleRob,我們是ag14分組')

    # Get handles for BubbleRob and its left and right motors
    res, bubbleRob = sim.simxGetObjectHandle(clientID, 'bubbleRob', sim.simx_opmode_oneshot_wait)
    res, leftMotor = sim.simxGetObjectHandle(clientID, 'leftMotor', sim.simx_opmode_oneshot_wait)
    res, rightMotor = sim.simxGetObjectHandle(clientID, 'rightMotor', sim.simx_opmode_oneshot_wait)

    # Set the initial motor velocities to zero
    sim.simxSetJointTargetVelocity(clientID, leftMotor, 0, sim.simx_opmode_oneshot_wait)
    sim.simxSetJointTargetVelocity(clientID, rightMotor, 0, sim.simx_opmode_oneshot_wait)

    # Control BubbleRob using the keyboard
    while True:
        # Set the motor velocities based on the arrow keys being pressed
        if keyboard.is_pressed('up'):
            print("up")
            sim.simxSetJointTargetVelocity(clientID, leftMotor, 7, sim.simx_opmode_oneshot_wait)
            sim.simxSetJointTargetVelocity(clientID, rightMotor, 7, sim.simx_opmode_oneshot_wait)
        elif keyboard.is_pressed('down'):
            print("down")
            sim.simxSetJointTargetVelocity(clientID, leftMotor, -7, sim.simx_opmode_oneshot_wait)
            sim.simxSetJointTargetVelocity(clientID, rightMotor, -7, sim.simx_opmode_oneshot_wait)
        elif keyboard.is_pressed('left'):
            print("left")
            sim.simxSetJointTargetVelocity(clientID, leftMotor, -4, sim.simx_opmode_oneshot_wait)
            sim.simxSetJointTargetVelocity(clientID, rightMotor, 4, sim.simx_opmode_oneshot_wait)
        elif keyboard.is_pressed('right'):
            print("right")
            sim.simxSetJointTargetVelocity(clientID, leftMotor, 4, sim.simx_opmode_oneshot_wait)
            sim.simxSetJointTargetVelocity(clientID, rightMotor, -4, sim.simx_opmode_oneshot_wait)
        else:
            # Stop the motors if no arrow keys are being pressed
            sim.simxSetJointTargetVelocity(clientID, leftMotor, 0, sim.simx_opmode_oneshot_wait)
            sim.simxSetJointTargetVelocity(clientID, rightMotor, 0, sim.simx_opmode_oneshot_wait)

        time.sleep(0.1)

- else:
    print('Failed connecting to remote API server')
```

圖. 6.8: 本地控制程式

6.3 遠端程式

如(圖.6.11)，須注意端口須改為19998。並且在leftMotor、rightMotor、bubbleRob輸入剛剛偵測的變值。

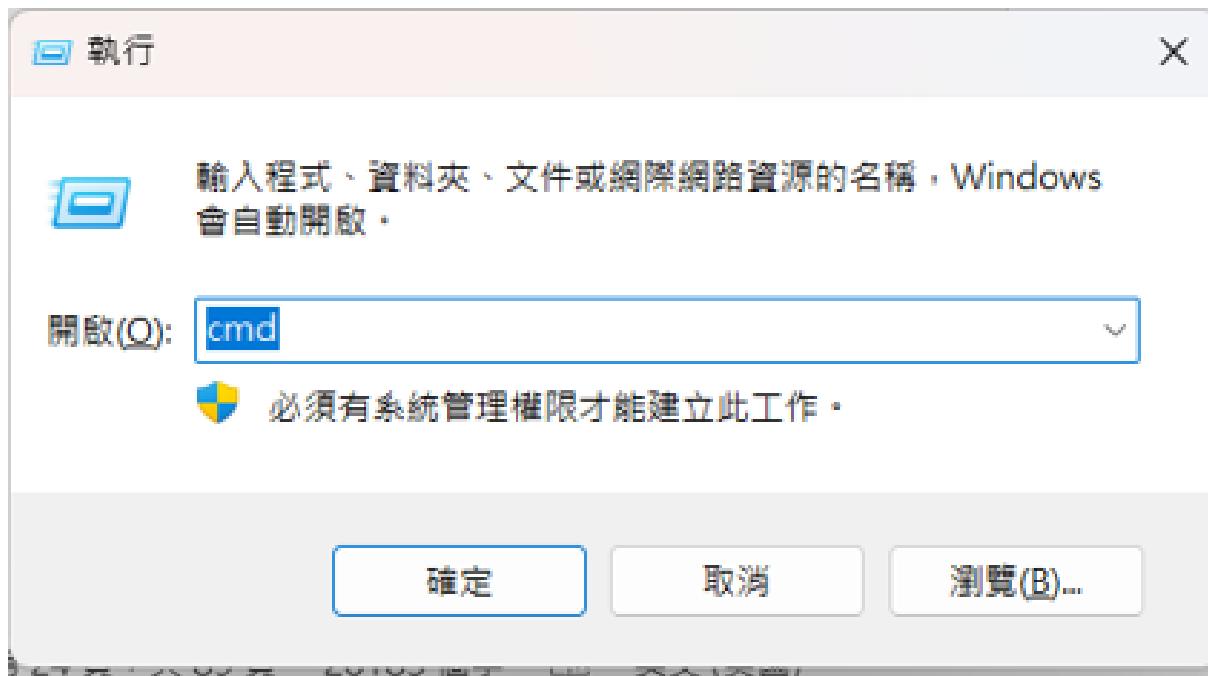


圖 6.9：查看本地 IP

```
本地連線 C:\WINDOWS\sys + v
媒體狀態 . . . . . : 媒體已中斷連線
連線特定 DNS 屬屬 . . . . . : TOTOLINK

無線區域網路介面卡 區域連線* 1:
媒體狀態 . . . . . : 媒體已中斷連線
連線特定 DNS 屬屬 . . . . . :

無線區域網路介面卡 區域連線* 2:
媒體狀態 . . . . . : 媒體已中斷連線
連線特定 DNS 屬屬 . . . . . :

乙太網路卡 乙太網路:
連線特定 DNS 屬屬 . . . . . :
IPv6 位址 . . . . . : 2001:b011:b809:9150:8dd5:2628:7b30:ee03
臨時 IPv6 位址 . . . . . : 2001:b011:b809:9150:4d8:b56f:3615:a619
連結-本機 IPv6 位址 . . . . . : fe80::882:dd57:9df4:b20%9
IPv4 位址 . . . . . : 192.168.1.113
子網路遮罩 . . . . . : 255.255.255.0
預設閘道 . . . . . : fe80::5ad5:6eff:fe00:ee53%9
                           192.168.1.1

乙太網路卡 藍牙網路連線:
媒體狀態 . . . . . : 媒體已中斷連線
連線特定 DNS 屬屬 . . . . . :

:C:\Users\User>
```

圖. 6.10: IP config

```

import sim
import sys
import time
import keyboard

# Connect to the simulation
sim.simxFinish(-1)
clientID = sim.simxStart('127.168.1.113', 19998, True, True, 5000, 5)

if clientID != -1:
    print('Hello,bubbleRob,我們是ag14分組')

# Get handles for BubbleRob and its left and right motors
res, bubbleRob = sim.simxGetObjectHandle(clientID, 'bubbleRob', sim.simx_opmode_oneshot_wait)
res, leftMotor = sim.simxGetObjectHandle(clientID, 'leftMotor', sim.simx_opmode_oneshot_wait)
res, rightMotor = sim.simxGetObjectHandle(clientID, 'rightMotor', sim.simx_opmode_oneshot_wait)
leftMotor = 28
rightMotor = 26
bubbleRob = 23
# Set the initial motor velocities to zero
sim.simxSetJointTargetVelocity(clientID, leftMotor, 0, sim.simx_opmode_oneshot_wait)
sim.simxSetJointTargetVelocity(clientID, rightMotor, 0, sim.simx_opmode_oneshot_wait)

# Control BubbleRob using the keyboard
while True:
    # Set the motor velocities based on the arrow keys being pressed
    if keyboard.is_pressed('up'):
        print("up")
        sim.simxSetJointTargetVelocity(clientID, leftMotor, 7, sim.simx_opmode_oneshot_wait)
        sim.simxSetJointTargetVelocity(clientID, rightMotor, 7, sim.simx_opmode_oneshot_wait)
    elif keyboard.is_pressed('down'):
        print("down")
        sim.simxSetJointTargetVelocity(clientID, leftMotor, -7, sim.simx_opmode_oneshot_wait)
        sim.simxSetJointTargetVelocity(clientID, rightMotor, -7, sim.simx_opmode_oneshot_wait)
    elif keyboard.is_pressed('left'):
        print("left")
        sim.simxSetJointTargetVelocity(clientID, leftMotor, -4, sim.simx_opmode_oneshot_wait)
        sim.simxSetJointTargetVelocity(clientID, rightMotor, 4, sim.simx_opmode_oneshot_wait)
    elif keyboard.is_pressed('right'):
        print("right")
        sim.simxSetJointTargetVelocity(clientID, leftMotor, 4, sim.simx_opmode_oneshot_wait)
        sim.simxSetJointTargetVelocity(clientID, rightMotor, -4, sim.simx_opmode_oneshot_wait)
    else:
        # Stop the motors if no arrow keys are being pressed
        sim.simxSetJointTargetVelocity(clientID, leftMotor, 0, sim.simx_opmode_oneshot_wait)
        sim.simxSetJointTargetVelocity(clientID, rightMotor, 0, sim.simx_opmode_oneshot_wait)

    time.sleep(0.1)

```

圖 . 6.11: 遠端控制程式

第七章 完成作業

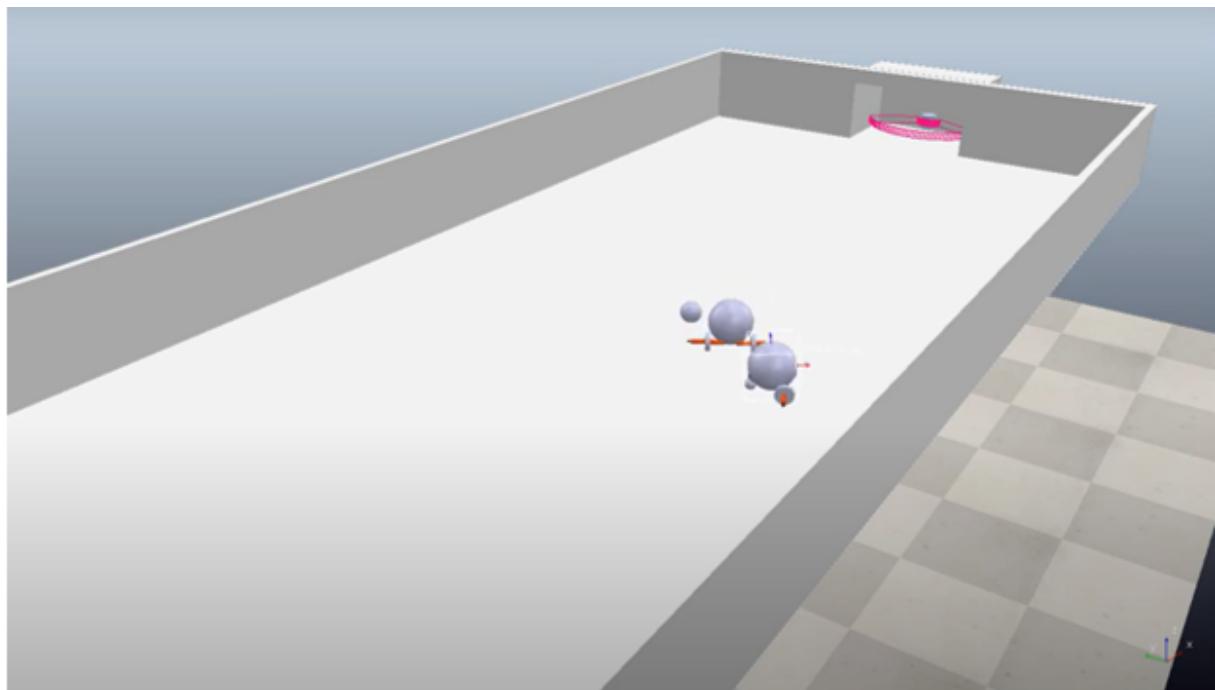


圖. 7.1: 完成作業