國 立 虎 尾 科 技 大 學
機 械 設 計 工 程 系

cd2023 2a-pj1ag4 分組報告

網際手足球機器人場景設計

Web-based Robot Soccer Scene Design

指導教授： 嚴 家 銘 老 師
班 級： 四 設 二 甲
學 生： 洪 偉 陞 (41023146)
夏 進 源 (41023148)
紀 閔 翔 (41023147)
施 建 昌 (41023143)
李 承 翰 (41023121)
林 建 維 (41023134)
呂 佳 柔 (41023104)
王 啟 騰 (41023112)

中華民國　　112　年　5　月

摘 要

本學期採取個人及團體分組來學習，團體實習目標為開發一款能在 web-based CoppeliaSim 場景中雙方或多方對玩的遊戲。pj3 為八人一組,，根據自選產品在期限內完成產品開發，在 w16 現場發表八人協同四週後所完成的產品，在 w17 各組採 OBS + Teams 以影片發表所完成的協同產品。

接續 pj2，各組須對雙輪車進行設計改良，以提升行進與對戰效率。採 CAD 進行場景與多輪車零組件設計後，轉入足球場景中以鍵盤 arrow keys 與 wasd 等按鍵進行控制，對陣雙方每組將有四名輪車球員，且每兩人在同一台電腦上操作，完成後各組須在分組網站中提供所有相關檔案下載連結，且提供線上分組簡報與分組 pdf 報告連結。

此專題是雙方利用各四台 BubbleRob 多輪車在一足球場景中進行對戰，雙方球門分別設有感測器。在規定時間內，每進一球即透過程式重與新往球場內隨機發球，接續賽局。模擬場景中還須配置 LED 計分板顯示比賽剩餘時間與比分，還需另外建立以機械轉盤傳動計分系統。在 CoppeliaSim 模擬環境中進行測試運用上的可行性並嘗試透過埠號供使用者觀看。

Abstract

This semester adopts both individual and group learning approaches. The group project involves developing a web-based game that allows two or more players to interact in a CoppeliaSim simulation environment. For Project 3 (pj3), the group consists of eight members who will collaborate to develop a product of their choice within a given deadline. In Week 16, the group will present the product they have developed over four weeks of collaboration. In Week 17, each group will use OBS + Teams to present their collaborative product through a video presentation.

Continuing from pj2, each group is required to design improvements for a two-wheeled vehicle to enhance its mobility and combat efficiency. Using CAD software, the groups will design the scene and components of the multi-wheel vehicle. The project will then transition to a soccer field scenario, where the vehicle will be controlled using keyboard arrow keys and WASD keys. Each group will have four vehicle players, with two players operating on the same computer. Upon completion, each group must provide download links for all relevant files on the group's website, as well as links to online group presentations and a PDF report.

This project involves a two-player battle using four BubbleRob multi-wheel vehicles in a soccer field scenario. Each player has their own goal equipped with sensors. Within a specified time frame, each goal scored triggers the program to reset and randomly kick off a new ball into the field, continuing the game. The simulation scene also includes an LED scoreboard to display the remaining time and score of the match. Additionally, a mechanical turntable-driven scoring system needs to be created. Feasibility testing and user observation will be conducted in the CoppeliaSim simulation environment, with the option for users to observe through port numbers.

目　錄

圖　目　錄

表　目　錄

第一章　前言

1.1 設計架構

此次 pj3 專題目標有建立場景中的計時器、球員外型及移動優化、添加球員擊球和翻車再起技能、進球後收集並隨機投下新的一顆球、建立以機械式轉盤傳動計分系統。由於目標繁多，需要組員間分工負責，在每個禮拜的協同中逐步完成 pj3 專題。

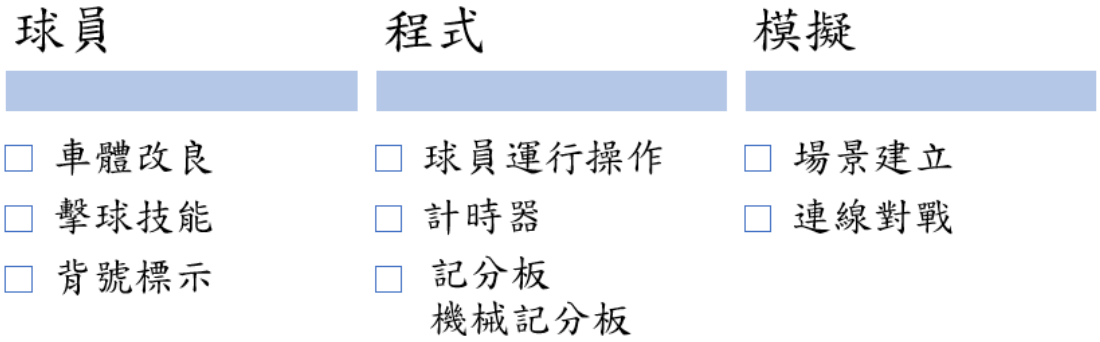

圖. 1.1: 設計目標圖

1.2 規則說明

類似於足球遊戲，一開始時球會置於場中央，遊戲開始後雙方即可以鍵盤操控機器人，透過防守敵方以及與隊友間的傳球推球至已方的球門得分。

遊戲規則如下：

1. 球觸碰到球門感測器即算得分。
2. 在十分鐘的鼻賽時間內，獲得最多分數的隊伍即獲勝。
3. 任一方進球得分後，隨機在場內投下新的球，雙方接續進行比賽。

第二章　球員製作過程

2.1 車體改良-運行

原本的車體為球形 bubbleRob，雖然造型簡單，卻會有容易翻車的問題。因此改為磚塊型，在前後添加兩顆輪子保持平衡。也將前進原理改為四輪驅動，使轉彎更為順暢合理。

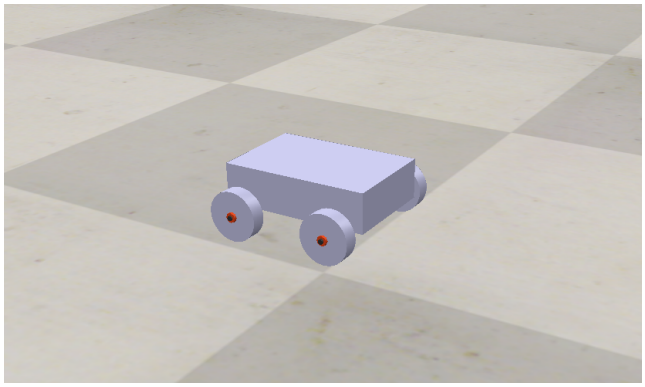

圖. 2.1: 磚塊型車體

2.2 車體改良-擊球

球員前端添加凸出的手部，球員本體在 CoppeliaSim 中用導入的開啟方式會產生抖動，因此改為加入物件 skin 並將本體隱藏。

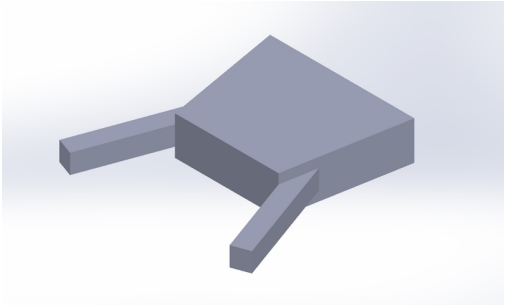

圖. 2.2: 球員 skin

2.3 車體改良-背號

由於兩隊各有四名球員，場上總共八名球員，為了能更清楚觀看及辨別球員，除了透過顏色區分隊伍，也需要讓每個球員添加背號。第一版的背號是直立式置於球員上方，實際遊玩時發現會有影響重心的問題。

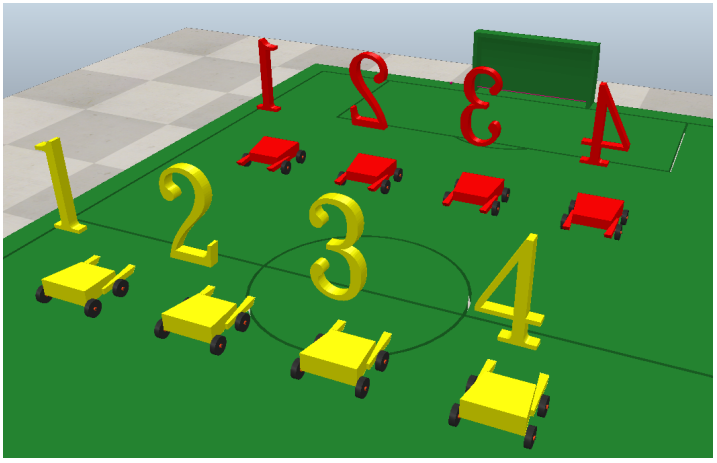

圖. 2.3: 第一版球員背號

於是第二版做了調整，參考實際賽車都將編號繪於車身，我們將背號改為平貼於車頂，解決了影響重心的問題。

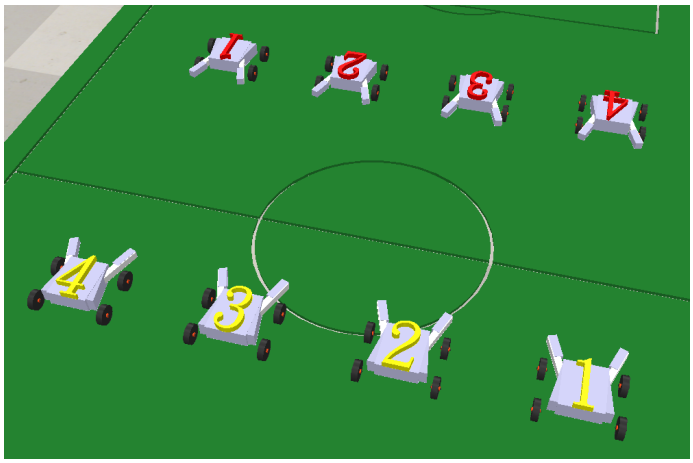

圖. 2.4: 第二版球員背號

第三章　球員程式碼

3.1 車體改良-操作

為了增加對戰的刺激性及操作的方便性，我們對球員的程式碼新增可以前後移動並同時左右移動，及添加倒地翻身的功能。

3.2 移動操作-輪子旋轉

```
def setBubbleRobVelocity(leftWheelVelocity1, rightWheelVelocity1,
leftWheelVelocity2, rightWheelVelocity2):
    leftMotor1 = sim.getObject('/a_player1/joint_lf')
    rightMotor1 = sim.getObject('/a_player1/joint_rf')
    leftMotor2 = sim.getObject('/a_player1/joint_lb')
    rightMotor2 = sim.getObject('/a_player1/joint_rb')
    sim.setJointTargetVelocity(leftMotor1, leftWheelVelocity1)
    sim.setJointTargetVelocity(rightMotor1, rightWheelVelocity1)
    sim.setJointTargetVelocity(leftMotor2, leftWheelVelocity2)
    sim.setJointTargetVelocity(rightMotor2, rightWheelVelocity2)
    #輸入四個變數分別給四個軸速度
```

圖. 3.1: 設定球員輪子旋轉

(圖.??) 設定一個 setVelocity 函數，接受四個參數：leftWheelVelocity1，rightWheelVelocity1，leftWheelVelocity2，rightWheelVelocity2，分別為左右前後輪速度。在函數內部使用 sim.getObject() 函數獲取左右前後輪的關節 joint ，並使用 sim.setJointTargetVelocity() 函數將各關節的目標速度設置為傳入的參數值，這樣做及可控制球員輪子速度，從而使球員移動。

3.3 移動操作-前輪方向

```
def setBubbleRobangel(a):
    brickRob= sim.getObject('/a_player1')
    angel = [-90*math.pi/180, a*math.pi/180, 0*math.pi/180]
    leftMotor = sim.getObject('/a_player1/joint_lf')
    rightMotor = sim.getObject('/a_player1/joint_rf')
    sim.setObjectOrientation(leftMotor, brickRob, angel)
    sim.setObjectOrientation(rightMotor, brickRob, angel)
    #輸入一個變數改變前輪方向
```

圖. 3.2: 設定球員前輪方向

(圖.??) 設定一個 setBubbleRobange1 函數，接受一個設置角度值得參數"a"，以函數 sim.getObject 獲取一個代表"a player" 模型的對象"brickRob"。計算角度值並傳入參數"a" 轉換為弧度值，將另兩個角度分別為固定角度值。在使用 sim.getObject() 獲取左右前輪的關節，以 sim.setObjectOrientation() 將關節地朝向設置為與 brickRob 相對應的角度，便可以給定角度旋轉並改變球員地朝向位置。

3.4 移動操作-左右轉向

```
def controlangel(y):
    if keyboard.is_pressed('a'):
        setangel(-y)
    elif keyboard.is_pressed('d'):
        setangel(y)
    else:
        setangel(0)
```

圖. 3.3: 控制球員左右轉向

(圖.??) controlange1 函數接受控制角度的函數"y"，檢查按鍵輸入，按下"a" 調用 setangel(-y) ，將傳入函數取相反值設成角度；按下"d" 則用 setangel(y)，傳入函數直接設成角度；無按下任何鍵則調用 setangel(0)，將角度設為零。

此程式碼根據按鍵輸入來控制角度值輸入，可以分別控制向左向右角度值。

3.5 移動操作-控制球員移動

```
def playercontrol(x,y):
    if keyboard.is_pressed('w'):
        setVelocity(x,x,x,x)
        controlangel(y)
    elif keyboard.is_pressed('s'):
        setVelocity(-x,-x,-x,-x)
        controlangel(y)
    elif keyboard.is_pressed('a'):
        setVelocity(-x,x,-x,x)
    elif keyboard.is_pressed('d'):
        setVelocity(x,-x,x,-x)
    elif keyboard.is_pressed('q'):
        # stop simulation
        sim.stopSimulation()
    else:
        setVelocity(0, 0, 0, 0)
        setangel(0)
```

圖. 3.4: wasd 移動球員

(圖.??) 以 playcontrol 函數定義參數"x" 和"y"，代表速度及角度。檢查按鍵輸入，如果按下"w" 將調用 setVelocity(x, x, x, x) 設置輪子速度、調用 controlangel(y) 控制角度。若為 setVelocity(-x, -x, -x, -x) 則設置為反向速度，按下"q" 則停止模擬。

根據鍵盤輸入 w, a, s, d 來控制球員前進、後退、左轉和右轉。

3.6 移動操作-維持速度

```
while True:
    if keyboard.is_pressed('shift'):
        playercontrol(v+4,a-20)
    else:
        playercontrol(v,a)
```

圖. 3.5: 控制球員速度

(圖.??) 此程式碼使用無限循環"while True" 迴圈，如果按下"shift" 會調用 playcontrol(v+4, a-20) 函數，使原有速度增加 4，角度減去 20。使用無線循環迴圈檢測鍵盤輸入，調整速度與角度。

3.7 球員改良-倒地翻身

```
elif keyboard.is_pressed('e'):
        floor= sim.getObject('/Floor')
        player = sim.getObject('/a_player1')
        a=sim.getObjectOrientation(player,floor)
        b=sim.getObjectPosition(player,floor)
        a[0]=0
        a[1]=0
        b[2]=b[2]+0.2
        sim.setObjectPosition(player,floor,b)
        sim.setObjectOrientation(player,floor,a)
```

圖. 3.6: 控制球員翻身

(圖.??) 定義如果按"e" 就執行，以 sim.getObject('/Floor') 獲取地板句柄、('/a player1') 球員句柄；sim.getObjectOrientation(player,floor) 獲取相對於地板的球員方向、Position(player,floor) 獲取相對於地板的球員位置，將數值存於變量"a" 及"b" 中。a[0]=0 將球員的 x 角度為 0，a[1]=0 將球員的 y 角度為 0，b[2]=b[2]+0.2 將球員的 z 位置上升 0.2。最後通過 sim.setObjectPosition(player,floor,b) 設定球員相對於地板的位置、Orientation(player,floor,a) 設定球員相對於地板的方向。

此段程式碼根據按下"e" 來執行將指定對象地朝向與位置進行修改，使球員翻倒後能夠再次翻身站起，繼續進行比賽。

第四章　計時器與記分板

4.1 摘要

完成球員設定後，接著說明場景中計時器設定與 LED 記分板和機械式轉盤記分板製作過程。

4.2 計時器

```
function sysCall_init()
    t2=0
    s0={1,1,1,0,1,1,1}
    s1={0,0,1,0,0,1,0}
    s2={1,0,1,1,1,0,1}
    s3={1,0,1,1,0,1,1}
    s4={0,1,1,1,0,1,0}
    s5={1,1,0,1,0,1,1}
    s6={1,1,0,1,1,1,1}
    s7={1,0,1,0,0,1,0}
    s8={1,1,1,1,1,1,1}
    s9={1,1,1,1,0,1,1}
    s={s0,s1,s2,s3,s4,s5,s6,s7,s8,s9}

    score(0,'a')
    score(0,'b')
    score(0,'c')
    score(0,'d')
end
```

圖. 4.1: 定義變量

(圖.??) 以函數 sysCall init 定義一些變量，將 t2 初始化為 0，s0 到 s9 由 0 到 1 組成的列表，用於後續計算。通過調用 score(0, 'a')、score(0, 'b')、score(0, 'c') 和 score(0, 'd') 來初始化名為 a、b、c、d 的得分。此程式碼做愈是在模擬環境初始化階段定義一些變量、列表及得分的部分。

(圖.??) score 函數接受參數"x"、"y"，使用一個循環重複數字 0 到 9，首先檢查"x" 是否等於當前數字，如果相等則執行內部循環。以 sim.getObject() 根據給定的字符串構建對象名稱並儲存在 part 變量中。再檢查 s[i+1][j+1] ，數字 i 在位置 j 上如果為 1 就使用 sim.setShapeColor() 將 part 對象的顏色設置為紅色（1, 0, 0），若非則將顏色設置為黑色（0, 0, 0）。

```
function score(x,y)
    for i=0 ,9,1 do
        if (x==i)then
            for j = 0,6,1 do
                local part = sim.getObject('./'..y..''..j..'')
                if (s[i+1][j+1]==1) then
                    sim.setShapeColor(part, nil, sim.
colorcomponent_ambient_diffuse, {1, 0, 0})
                else
                    sim.setShapeColor(part, nil, sim.
colorcomponent_ambient_diffuse, {0, 0, 0})
                end
            end
        end
    end
end
```

圖. 4.2: 數字顏色

根據輸入的數字"x"、"y" 設置對應數字的顏色，以數字的模式在模擬環境中的相對應位置設置不同的顏色。

```
function sysCall_actuation()

    simulationTime = sim.getSimulationTime()/4
    simulationTime1=600-simulationTime

    t1=math.floor((simulationTime1/600)%10)
    score(t1,'a')
    t2=math.floor((simulationTime1/60)%10)
    score(t2,'b')
    t3=math.floor((simulationTime1/10)%6)
    score(t3,'c')
    t4=math.floor(simulationTime1%10)
    score(t4,'d')

    if (simulationTime1<1)then
        sim.pauseSimulation()
    end

    p=sim.getSimulationState()
    if(p==22)then
        score(0,'a')
        score(0,'b')
        score(0,'c')
        score(0,'d')
    end


end
```

圖. 4.3: 更改數字形狀的顏色

(圖.??) sysCall actuation 在模擬時被調用，首先藉由 sim.getSimulationTime() 獲取當前時間並除以 4，儲存在變量 simulationTime 中。以 simulation-Time 計算"t1"、"t2"、"t3" 及"t4"，分別表示當前時間的不同部分，所得

值通過 simulationTime1 除以相對應數字，並取整數部分。

再來以 score() 將計算得到的值分別與"a"、"b"、"c"、"d" 一起傳遞，可以根據時間的變化來更新相應的數字形狀的顏色。

根據模擬時間的變化更新數字形狀的顏色，在特定條件下暫停模擬或重置顯示的顏色。

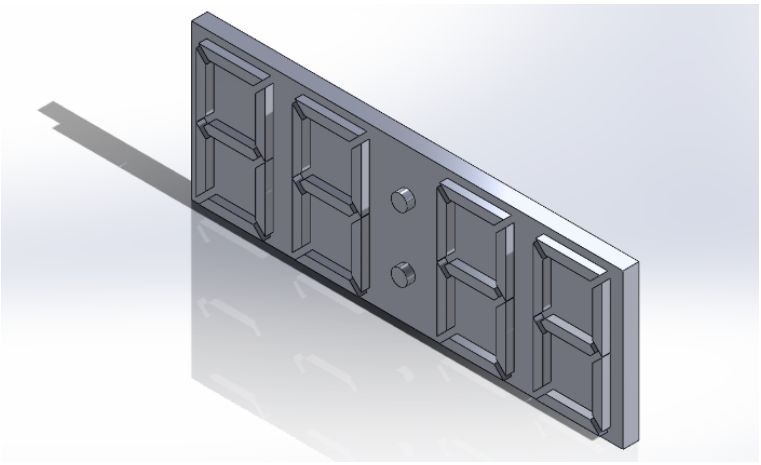

圖. 4.4: 計時器

4.3 LED 記分板

```
function sysCall_init()
    score1 = 0

    sensor = sim.getObject('./sensor')
    xml = [[
        <ui title="Scoreboard" closeable="false" resizable="false" style="plastique">
        <label text="Score:" style="* {background-color: #808080; color: #000000; font-size: 40px; font-weight: bold; padding: 5px; border-radius: 5px; }" id="10"/>
        <label text="0" style="* {background-color: #FFF; color: #000000; font-size: 40px; font-weight: bold; padding: 5px; border-radius: 5px;}" id="30"/>

        </ui>
    ]]
    ui = simUI.create(xml)
    simUI.setPosition(ui, 0,0, true)
    bubbleRob = sim.getObject('/bubbleRob')
    ball = sim.getObject('/ball')
    bubbleRob2 = sim.getObject('/bubbleRob2')
    initialPosition = sim.getObjectPosition(bubbleRob, -1)
    initialOrientation = sim.getObjectOrientation(bubbleRob, -1)
    initialPosition2 = sim.getObjectPosition(bubbleRob2, -1)
    initialOrientation2 = sim.getObjectOrientation(bubbleRob2, -1)
    initialballPosition = sim.getObjectPosition(ball, -1)
    initialballOrientation = sim.getObjectOrientation(ball, -1)

end


function sysCall_actuation()
    --simUI.setLabelText(ui, 30, tostring(sim.getFloatSignal("myVariable")))
    result=sim.readProximitySensor(sensor)
    if(score1<5)then
        if(result>0)then
            score2 = score1+1
            simUI.setLabelText(ui, 30, tostring(score2))

            sim.setObjectPosition(bubbleRob, -1, initialPosition)
            sim.setObjectOrientation(bubbleRob, -1, initialOrientation)
            sim.setObjectPosition(bubbleRob2, -1, initialPosition2)
            sim.setObjectOrientation(bubbleRob2, -1, initialOrientation2)
            sim.setObjectPosition(ball, -1, initialballPosition)
            sim.setObjectOrientation(ball, -1, initialballOrientation)
            score1=score2
        end
    else
        sim.pauseSimulation()
    end
end
```

圖. 4.5: 記分板

透過 Lua 程式建立記分板，設計樣式，給予偵測定義。

```
1   function sysCall_init()
2       score1 = 0
3   #Lua程式語言中，定義了一個名為"sysCall_init"的函數。函數內部，定義了一個名為"score1"的變數，初
    始值為0
4       sensor = sim.getObject('./sensor')
5   #定義了一個變數"sensor"，並且透過函數"sim.getObject"取得
```

圖. 4.6: sensor

(圖.4.6)sysCall init 的函式建立了一個名為 score1 的變數，並將它的值設定為 0 使用 sim.getObject 方法來獲取一個名為 sensor 的物件，並將它儲存在 sensor 變數中。

```
7   #變數名稱是"xml"。這個變數定義了一個包含XML內容的字串
8   —   xml = [[
9           <ui title="Scoreboard" closeable="false" resizable="false" style="plastique">
10          <label text="Score:" style="* {background-color: #808080; color: #000000;
    font-size: 40px; font-weight: bold; padding: 5px; border-radius: 5px; }" id="10"/>
11          <label text="0" style="* {background-color: #FFF; color: #000000; font-size: 40px;
    font-weight: bold; padding: 5px; border-radius: 5px;}" id="30"/>
12
13          </ui>
14      ]]
15  #Line9-用戶界面（UI），標題為"Scoreboard"，且不可關閉和改變大小，風格設定為"plastique"
16  #Line10-定義了一個標籤（label），顯示文字為"Score:"。風格設定：背景色為"#808080"，灰色；標籤
    的文字顏色為"#000000"，黑色；標籤的字體大小為"40px"，即40個像素；字體粗細為"bold"，即加粗；
    內邊距為"5px"，即內容和邊框的距離；邊框半徑為"5px"，即邊框的圓角程度；id為"10"，用於後續對這
    個標籤進行操作
17  #Line11-建立了一個標籤元素，並設置文字為 "0"。樣式屬性設置了背景顏色為白色
    (#FFF)，文字顏色為黑色 (#000000)，字體大小為 40px，字體粗細為粗體，padding 和
    border-radius 屬性用於對標籤進行視覺化設置， id 屬性設置為 "30"
```

圖. 4.7: xml

(圖.4.7存儲在名為 xml 的變數中的 XML 代碼，它描述了一個顯示得分的視窗界面。ui 標籤中，有兩個 label 標籤，分別用來顯示"Score:"和得分。

```
19      ui = simUI.create(xml)
20      simUI.setPosition(ui, 0,0, true)
21      bubbleRob = sim.getObject('/bubbleRob')
22      ball = sim.getObject('/ball')
23      bubbleRob2 = sim.getObject('/bubbleRob2')
24      initialPosition = sim.getObjectPosition(bubbleRob, -1)
25      initialOrientation = sim.getObjectOrientation(bubbleRob, -1)
26      initialPosition2 = sim.getObjectPosition(bubbleRob2, -1)
27      initialOrientation2 = sim.getObjectOrientation(bubbleRob2, -1)
28      initialballPosition = sim.getObjectPosition(ball, -1)
29      initialballOrientation = sim.getObjectOrientation(ball, -1)
30
31  end
32  #定義了一個 UI 元素，其內容為 xml 變數中的
    XML代碼,並將其顯示在模擬器中。接下來，定義了幾個模擬器對象的變數，分別是 bubbleRob、ball 和
    bubbleRob2，分別代表 BubbleRob 機器人和兩個球。然後，通過 sim.getObjectPosition() 和
    sim.getObjectOrientation() 函數來獲取 BubbleRob 和球的初始位置和方向
```

圖. 4.8: ui

(圖.4.8建立使用者介面 (ui)，然後設定該介面的位置為 (0, 0)。sim.getObject 函式從仿真場景中取得了三個物體，sim.getObjectPosition 和 sim.getObjectOrientation 函式取得了這些物體的初始位置和初始方向，這段程式碼是用來建立仿真場景中物體的初始位置和初始方向。

```
34  function sysCall_actuation()
35      --simUI.setLabelText(ui, 30, tostring(sim.getFloatSignal("myVariable")))
36      result=sim.readProximitySensor(sensor)
37  #在每個仿真週期中，sysCall_actuation函數會被呼叫一次，使用sim.readProximitySensor讀取與感測
    器相連的感測器元件的當前值，並將其存儲在result變量中
```

圖. 4.9: syscall

(圖.4.9sysCall-actuation 的函式使用 sim.readProximitySensor 函式讀取一個接近傳感器的數據，並將結果儲存在 result 變數中。

```
39      if(score1<5)then
40          if(result>0)then
41              score2 = score1+1
42              simUI.setLabelText(ui, 30, tostring(score2))
43
44              sim.setObjectPosition(bubbleRob, -1, initialPosition)
45              sim.setObjectOrientation(bubbleRob, -1, initialOrientation)
46              sim.setObjectPosition(bubbleRob2, -1, initialPosition2)
47              sim.setObjectOrientation(bubbleRob2, -1, initialOrientation2)
48              sim.setObjectPosition(ball, -1, initialballPosition)
49              sim.setObjectOrientation(ball, -1, initialballOrientation)
50              score1=score2
51          end
52      else
53  #如果score1小於5，而且偵測到感測器的訊號（result>0），則將分數加1（score2 =
    score1+1），並在UI上更新分數（simUI.setLabelText(ui, 30,
    tostring(score2))）。程式碼會將所有物體的位置和方向重設為初始值，分別是BubbleRob、BubbleRob2
    和球（ball）。最後將score1設為新的score2，以便下一次感測器觸發時更新分數。如果score1已經大於或
    等於5，則什麼都不做
```

圖. 4.10: if-else

(圖.4.10這段程式碼的主要作用是檢查分數是否小於 5 分，如果 score1 的值小於 5，且 result 大於 0，則將分數加 1 並在 UI 上更新分數。如果 score1 的值已經達到或超過了 5 分，則會跳過。

```
55              sim.pauseSimulation()
56          end
57      end
58      #用於暫停 CoppeliaSim 模擬運行。當這個函數被調用時，模擬會暫停在當前時間點
```

圖. 4.11: pause

(圖.4.11檢查分數是否小於 5 分。如果分數達到 5 分或更高，它會暫停仿真並結束程式執行。

4.4 機械式轉盤記分板

除了採用 LED 顯示計分外，另外建立以機械轉盤傳動計分系統，納入每按一下"i" 轉盤及順時鐘旋轉 36 度。

```
function sysCall_init()


    sensor = sim.getObject('/sensor1')
    sensor = sim.getObject('/sensor2')
    a1=sim.getObject('/scorebord/board_a/joint[1]')
    a2=sim.getJointTargetPosition(a1)
    b1=sim.getObject('/scorebord/board_b/joint[1]')
    b2=sim.getJointTargetPosition(b1)
end



function sysCall_actuation()
    sensor1 = sim.getObject('/sensor1')
    sensor2 = sim.getObject('/sensor2')
    result1=sim.readProximitySensor(sensor1)
    result2=sim.readProximitySensor(sensor2)
    --if (result1>0)then


    if (result1>0)then
        a3=a2+36*math.pi/180
        sim.setJointTargetPosition(a1,a3)
        a2=a3
    elseif (result2>0)then
        b3=b2+36*math.pi/180
        sim.setJointTargetPosition(b1,b3)
        b2=b3

    end


end
```

圖. 4.12: 機械記分板

```
function sysCall_init()


    sensor = sim.getObject('/sensor1')
    sensor = sim.getObject('/sensor2')
    a1=sim.getObject('/scorebord/board_a/joint[1]')
    a2=sim.getJointTargetPosition(a1)
    b1=sim.getObject('/scorebord/board_b/joint[1]')
    b2=sim.getJointTargetPosition(b1)
end
```

圖. 4.13: 獲取變量

(圖.4.13 使用 sim.getObject() 獲取 /sensor1 存入變量"sensor"、獲取 /scorebord/board a/joint[1] 的關節存入變量"a1"，以 sim.getJointTargetPosition() 獲取"a1" 目標位置存入變量"a2"，獲取變量"b1" 目標位置存入變量"b2"。

```
function sysCall_actuation()
    sensor1 = sim.getObject('/sensor1')
    sensor2 = sim.getObject('/sensor2')
    result1=sim.readProximitySensor(sensor1)
    result2=sim.readProximitySensor(sensor2)
    --if (result1>0)then


    if (result1>0)then
        a3=a2+36*math.pi/180
        sim.setJointTargetPosition(a1,a3)
        a2=a3
    elseif (result2>0)then
        b3=b2+36*math.pi/180
        sim.setJointTargetPosition(b1,b3)
        b2=b3

    end


end
```

圖. 4.14: 角度偏移

(圖.4.14 sim.readProximitySensor() 函數獲取近接傳感器"sensor1"、"sensor2" 的結果，存入變量"result1"、"result2"，如果大於 0 則執行：關節"a1" 的目標位置為當当前位置"a2" 加上 36 度的角度偏移，並將"a2" 更新為"a3" 以便下次執行使用。

此程式碼根據近接傳感器的檢測結果來控制關節運動，使機械式轉盤隨比分計分。