國 立 虎 尾 科 技 大 學

機 械 設 計 工 程 系

cd2023 2a-pj1ag6 分組報告

網際手足球場景設計

Web-based Football Scene Design

指導教授： 嚴 家 銘 老 師

班 級： 四 設 二 甲

學 生： 呂 佳 柔 (41023104)

王 啟 騰 (41023112)

中華民國 112 年 3 月

國立虎尾科技大學　機械設計工程系
分組報告製作合格認可證明

分組報告製作修習學生：　四設二甲 41023104 呂佳柔
四設二甲 41023112 王啟騰

分組報告題目　：網際手足球場景設計

經評量合格，特此證明

評 審 委 員：＿＿＿＿＿＿＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿＿＿＿＿＿＿

指 導 老 師：＿＿＿＿＿＿＿＿＿＿＿＿＿＿

中 華 民 國 一 一 二 年四 月 十 七 日

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。

除了設計 bubbleRob 機器人、繪製球場、設定感測器及寫能讓雙方對打的程式碼外，組員間也需要共同維護、分工合作。

此專題是利用兩台 BubbleRob 雙輪車在一足球場景中進行對戰，雙方球門分別設有感測器，各有一名 BubbleRob 負責運球。在規定時間內，每進一球即透過程式重新從球場中線發球，重啟賽局。模擬場景中還須配置計分板顯示比賽剩餘時間與比分。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, individual and group grouping will be adopted for learning. The group practical goal is to develop a game that can be played by two or more parties in a web-based CoppeliaSim scene.

In addition to designing the bubbleRob robot, drawing the playing field, setting up sensors, and writing code that enables both parties to play, team members also need to maintain and collaborate on tasks.

This project involves using two BubbleRob wheeled robots to compete in a soccer field scene, with sensors installed on each goal post. Each team has one BubbleRob responsible for ball handling. Within a specified time, each goal scored will trigger a program to reset the game from the center of the field, restarting the match. A scoreboard is also configured in the simulation scene to display the remaining time and score. The feasibility of testing and using the project in the CoppeliaSim simulation environment will be explored, and users will be able to view the simulation via port number.

目　錄

圖 目 錄

表　目　錄

第一章　前言

1.1 設計架構

此次專題運用到三大類主題，利用 Solidworks 繪製球場，在 CoppeliaSim 中以虛擬環境創建、模擬和分析複雜的機器人系統，以 Lua 程式編寫對機器人的操控、設計遊玩規則。

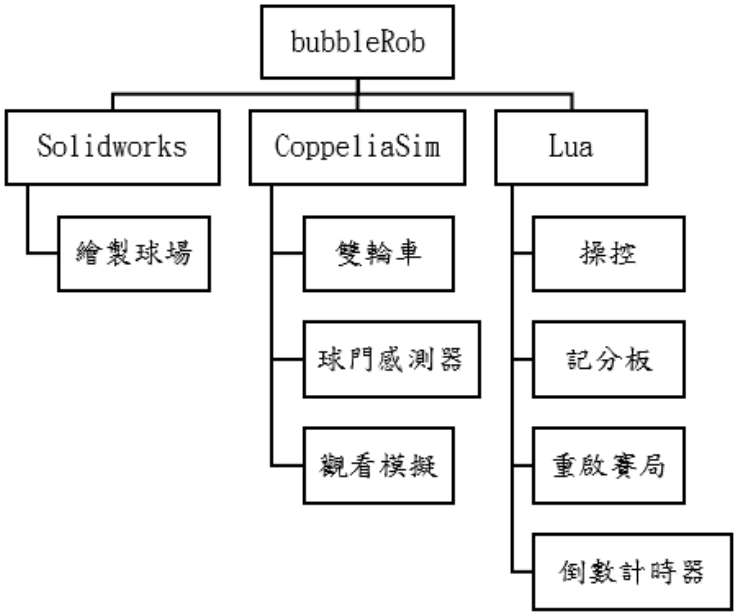

圖. 1.1: 設計架構圖

1.2 規則說明

類似於足球遊戲，一開始時球會置於場中央，遊戲開始後兩方即可以鍵盤操控機器人推球至已方的球門得分。

遊戲規則如下：

1. 球觸碰到情們感測器即算得分。
2. 最快獲得 5 分的隊伍即獲勝。
3. 任一方進球得分後，遊戲會重置，雙方回到場中央重新開球。

第二章　Lua/ Python 程式碼

2.1 Robot 控制

```python
# pip install pyzmq cbor
from zmqRemoteApi import RemoteAPIClient
import keyboard

client = RemoteAPIClient('localhost', 23000)

print('Program started')
sim = client.getObject('sim')
sim.startSimulation()
print('Simulation started')

def setBubbleRobVelocity(leftWheelVelocity, rightWheelVelocity):
    leftMotor = sim.getObject('/leftMotor2')
    rightMotor = sim.getObject('/rightMotor2')
    sim.setJointTargetVelocity(leftMotor, leftWheelVelocity)
    sim.setJointTargetVelocity(rightMotor, rightWheelVelocity)

'''
# Example usage 1:
setBubbleRobVelocity(1.0, 1.0)
time.sleep(2)
setBubbleRobVelocity(0.0, 0.0)
'''
# use keyborad to move BubbleRob

while True:
    if keyboard.is_pressed('up'):
        setBubbleRobVelocity(3.0, 3.0)
    elif keyboard.is_pressed('down'):
        setBubbleRobVelocity(-3.0, -3.0)
    elif keyboard.is_pressed('left'):
        setBubbleRobVelocity(-3.0, 3.0)
    elif keyboard.is_pressed('right'):
        setBubbleRobVelocity(3.0, -3.0)
    elif keyboard.is_pressed('q'):
        # stop simulation
        sim.stopSimulation()
    else:
        setBubbleRobVelocity(0.0, 0.0)
```

圖. 2.1: 本機操控

以 Lua 程式在模擬環境中控制機器人運行。

```python
# pip install pyzmq cbor 使用pip安裝了兩個Python模組：pyzmq和cbor
from zmqRemoteApi import RemoteAPIClient
import keyboard
#用於與遠程API服務器進行通信， 接收和處理用戶鍵盤輸入的事件

client = RemoteAPIClient('localhost', 23000)
#創建了一個名為"client"的遠程API客戶端對象， 傳遞了兩個參數：一個IP地址和一個端口號
```

圖. 2.2: pip install

(圖.2.2) 使用 RemoteAPIClient 來連接遠端的 API 服務器，以及使用 keyboard 函式庫來控制鍵盤。

```
9   print('Program started') #使用print函數輸出提示程式已經啟動
10  sim = client.getObject('sim')
11  sim.startSimulation()
12  print('Simulation started')
13  #Line10-通過調用client對象的getObject方法，從遠程API服務器上獲取了一個名為“sim”的對象，並將其賦值給變量“sim”
14  #Line11-調用“sim”對象的startSimulation方法，開始了模擬運行
15  #Line12-使用print函數輸出提示模擬已經啟動
```

圖. 2.3: 仿真環境

(圖.2.3) 先使用 print 函式輸出一個提示訊息，表示程式已經啟動。接著，使用 RemoteAPIClient 物件的 getObject 方法來從遠端 API 服務器上獲取一個名為 sim 的物件。再呼叫 sim 物件的 startSimulation 方法來啟動模擬器的運行。最後，使用 print 函式輸出模擬器已經啟動。

```
17  def setBubbleRobVelocity(leftWheelVelocity, rightWheelVelocity):
18      leftMotor = sim.getObject('/leftMotor2')
19      rightMotor = sim.getObject('/rightMotor2')
20      sim.setJointTargetVelocity(leftMotor, leftWheelVelocity)
21      sim.setJointTargetVelocity(rightMotor, rightWheelVelocity)
22
    #Line14-"setBubbleRobVelocity”的函數，接受兩個參數leftWheelVelocity和rightWheelVelocity，分別表示BubbleRob機器人左輪和右輪的速度
23      #Line15-調用sim對象的"getObject"方法從遠程API服務器上獲取控制左輪和右輪引擎
24
    #Line20-調用sim對象的"setJointTargetVelocity"方法，將leftMotor和rightMotor對象的目標速度分別設置為leftWheelVelocity和rightWheelVelocity
```

圖. 2.4: 控制左右輪速度

(圖.2.4)leftWheelVelocity 和 rightWheelVelocity，分別表示左輪和右輪的速度。sim.getObject 獲取 leftMotor2 和 rightMotor2 馬達物件。sim.setJointTargetVelo 方法來設定它們的目標速度。此段程式碼可以控制機器人的移動。

```
'''
# Example usage 1:
setBubbleRobVelocity(1.0, 1.0)
time.sleep(2)
setBubbleRobVelocity(0.0, 0.0)
'''
#Line28-"setBubbleRobVelocity"函數設置機器人的左輪和右輪速度為1.0，time.sleep(2)函數暫停程序執行2秒，讓機器人運動。將機器人的左輪和右輪速度都設置為0，使機器人停止運動

# use keyborad to move BubbleRob
#while迴圈會不斷檢查按鍵事件並執行相應的操作
```

圖. 2.5: 機器人運動

```
while True:
    if keyboard.is_pressed('up'):
        setBubbleRobVelocity(3.0, 3.0)
    elif keyboard.is_pressed('down'):
        setBubbleRobVelocity(-3.0, -3.0)
    elif keyboard.is_pressed('left'):
        setBubbleRobVelocity(-3.0, 3.0)
    elif keyboard.is_pressed('right'):
        setBubbleRobVelocity(3.0, -3.0)
    elif keyboard.is_pressed('q'):
        # stop simulation
        sim.stopSimulation()
    else:
        setBubbleRobVelocity(0.0, 0.0)

#Line38-"keyboard.is_pressed"函數檢查當前是否有按鍵事件。如果檢測到某個按鍵被按下，就會調用"setBubbleRobVelocity"函數。控制機器人的運動方向和左右輪速度
#Line46-q鍵被按下，就會調用"sim.stopSimulation"函數停止仿真
#如果沒有任何按鍵事件發生，左輪和右輪速度都設置為0，使機器人停止運動
```

圖. 2.6: 偵測按鍵

2.2 聯機控制

```
1  # pip install pyzmq cbor
2  from zmqRemoteApi import RemoteAPIClient
3  import keyboard
4
5  client = RemoteAPIClient('192.168.50.227', 23000)
```

圖. 2.7: 聯機參數

(圖.2.5) 指定了一個 IP 位址 192.168.50.227 和一個端口號 23000 作為連接遠端 API 服務器的方式。RemoteAPIClient 可以從遠端 API 服務器獲取數據或控制遠端機器人的運動。

2.3 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. 2.8: 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. 2.9: sensor

(圖.2.7)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. 2.10: xml

(圖.2.8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. 2.11: ui

(圖.2.9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. 2.12: syscall

(圖.2.10sysCall-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. 2.13: if-else

(圖.2.11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. 2.14: pause

(圖.2.12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。

參考文獻

[1] https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c

[2] https://towardsdatascience.com/derivative-of-the-sigmoid-function-536880cf918e

[3] http://www.incompleteideas.net/book/RLbook2020.pdf

[4] https://medium.com/change-the-world-with-technology/policy-gradient-181d43a24cf5

[5] https://livebook.manning.com/book/grokking-deep-reinforcement-learning/chapter-11/v-11/38

[6] http://ukko.life.nctu.edu.tw/ u0517047/usage.html

[7] https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html

[8] https://uupgrade.medium.com/python-那些年我們一起玩過的遊戲-三-打磚塊-d89b648896ca

[9] https://cvfiasd.pixnet.net/blog/post/275774124-深度學習激勵函數介紹

[10] https://www.coppeliarobotics.com/helpFiles/

[11] https://hackernoon.com/the-reason-behind-moving-in-the-direction-opposite-to-the-gradient-f9566b95370b

[12] https://ruder.io/optimizing-gradient-descent/

[13] https://zh.wikipedia.org/wiki/HSL 和 HSV 色彩空間

[14] https://gist.github.com/karpathy/a4166c7fe253700972fcbc77e4ea32c5#file-pg-pong-py

[15] https://github.com/schinger/pong_ actor-critic/blob/master/pg-pong-ac.py

[16] https://gist.github.com/etienne87/6803a65653975114e6c6f08bb25e1522

附錄

LaTeX

LaTex 為一種程式語言，支援標準庫 (Standard Libraries) 和外部程式庫 (External Libraries)，不過與一般程式語言不同的是，它可以直接表述 Tex 排版結構，類似於 PHP 之於 HTML 的概念。但是直接撰寫 LaTex 仍較複雜，因此可以藉由 Markdown 這種輕量的標註式語言先行完成文章，再交由 LaTex 排版。此專題報告採用編輯軟體為 LaTeX，綜合對比 Word 編輯方法，LaTeX 較為精準正確、更改、製作公式等，以便符合規範、製作。

表. 1: 文字編輯軟體比較表

	相容性	直觀性	文件排版	數學公式	微調細部
LaTeX	√		√	√	√
Word		√			√

- 特點:

1. 相容性：以 Word 為例會有版本差異，使用較高版本編輯的文件可能無法以較低的版本開啟，且不同作業系統也有些許差異；相比 LaTeX 可以利用不同編譯器進行編譯，且為免費軟體也可移植至可攜系統內，可以搭配 Github 協同編譯。
2. 文件排版：許多規範都會要求使用特定版型，使用文字編譯環境較能準確符合規定之版型，且能夠大範圍的自定義排定所需格式，並能不受之後更改而整體格式變形。
3. 數學公式呈現：LaTex 可以直接利用本身多元的模組套件加入、編輯數學公式，在數學推導過程能夠快速的輸入自己需要的內容即可。
4. 細部調整：在大型論文、報告中有多項文字、圖片、表格，需要調整細部時，要在好幾頁中找尋，而 LaTeX 可以分段章節進行編譯，再進行合併處理大章節。

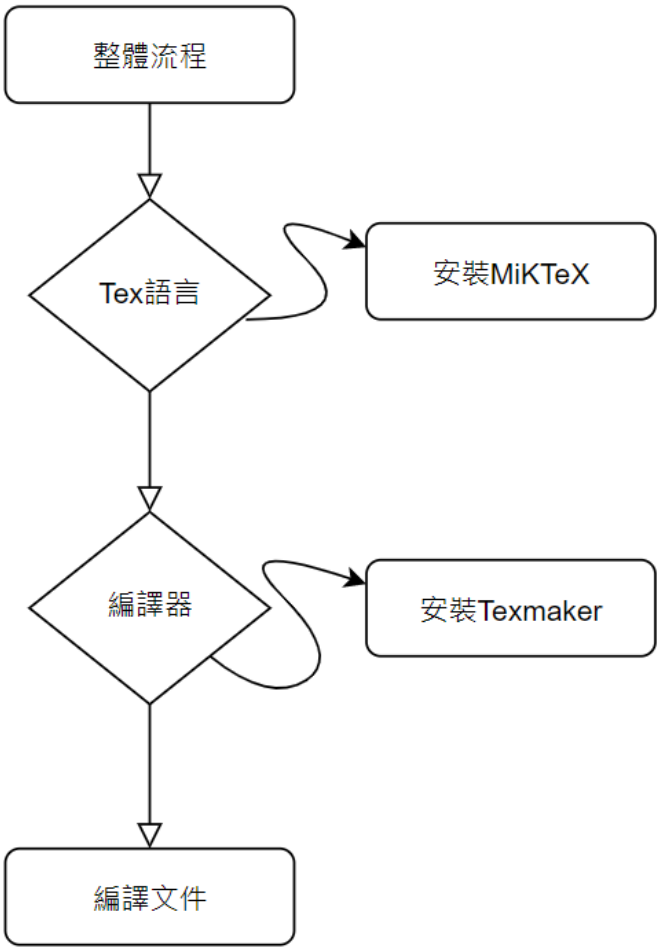

圖. 15: 編譯流程

FFmpeg

FFmpeg 是一個開放原始碼的自由軟體，可以對音訊和視訊進行多種格式的錄影、轉檔、串流功能。在專題訓練過程中透過 FFmpeg 的視訊錄製的功能記錄對打影像來了解實際訓練狀況。

作者簡介

姓名　　：王啟騰
學號　　：41023112
畢業學校：臺北市立松山高級工農職業學校
　　機械科

姓名　　：呂佳柔
學號　　：41023104
畢業學校：臺中市立臺中工業高級中等學校
　　機械製圖科