協同產品設計實習

Webot 模擬系統之投籃遊戲

姓　　名：王樟皓
學　　號：41023114
指導教授：嚴家銘 教授

1.基本資訊

在老師的範本中，是將 sensor 以圓球形置於球框，並且 radius 設 0.2(圖 1)

接著是 sensor 的很重要的一環，lookup table 是用來告訴 Webots 模擬器：感測器讀到的值要怎麼對應成實際回傳的數值。
如圖 3
1.距離是 0 公尺時，感測器回傳 1000，有 1%雜訊
2.距離是 0.12 公尺時，感測器回傳 620，有 1%雜訊
依此類推

下方(圖 2)的 type 主要是對紅色物體敏感、不會畫出紅點、會忽略透明物體

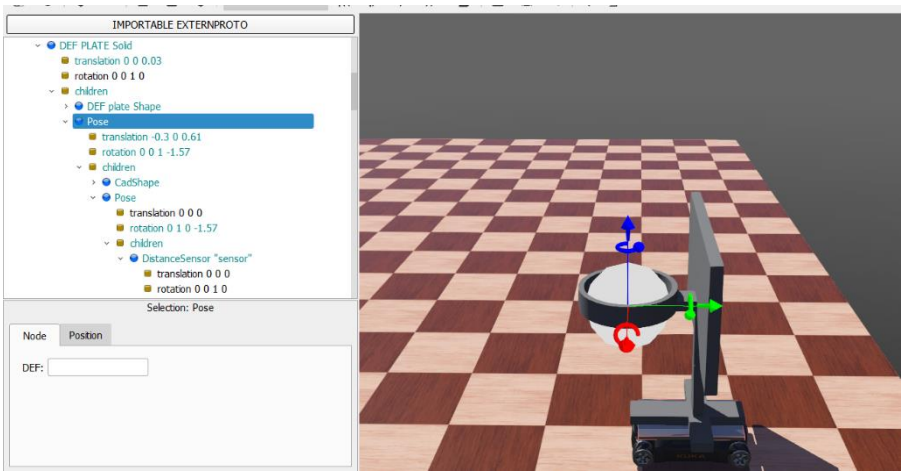

圖 1

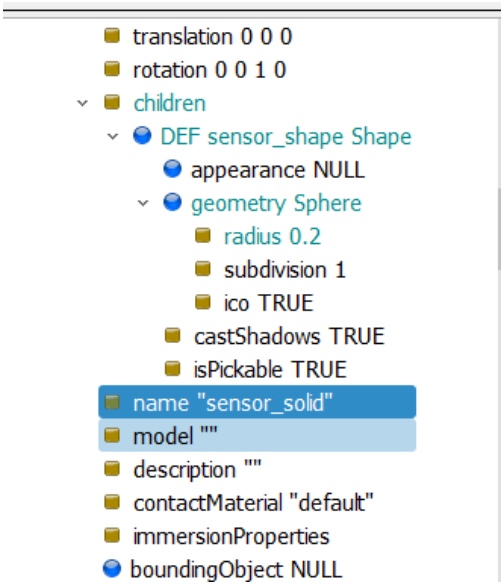

圖 2

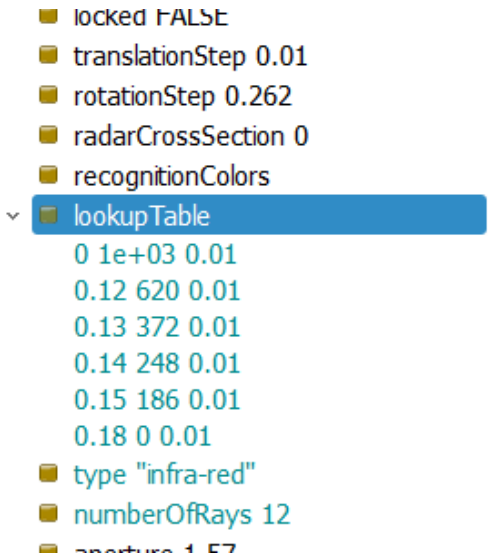

圖 3

Center of Mass 是質心，如果質心太高或偏一邊，在移動或受到力量時，就更容易翻倒。

Inertia Matrix 慣性矩陣簡單解釋：

它是一個 3x3 的矩陣（數學上叫「張量」），裡面包含了物體繞 x、y、z 三個方向旋轉時的轉動慣量。

如果你是圓盤或輪子那種簡單形狀，就只要一個值；但對於複雜 3D 形狀，就需要整個矩陣來描述。

| Ixx 0 0 |

| 0 Iyy 0 |

| 0 0 Izz |

其中：

Ixx 是繞 x 軸的轉動慣量

Iyy 是繞 y 軸的

Izz 是繞 z 軸的

而這數值投球機跟球框是一樣的，因為他們用的 youbot 是一樣的，輪子就是一樣的

如圖 4

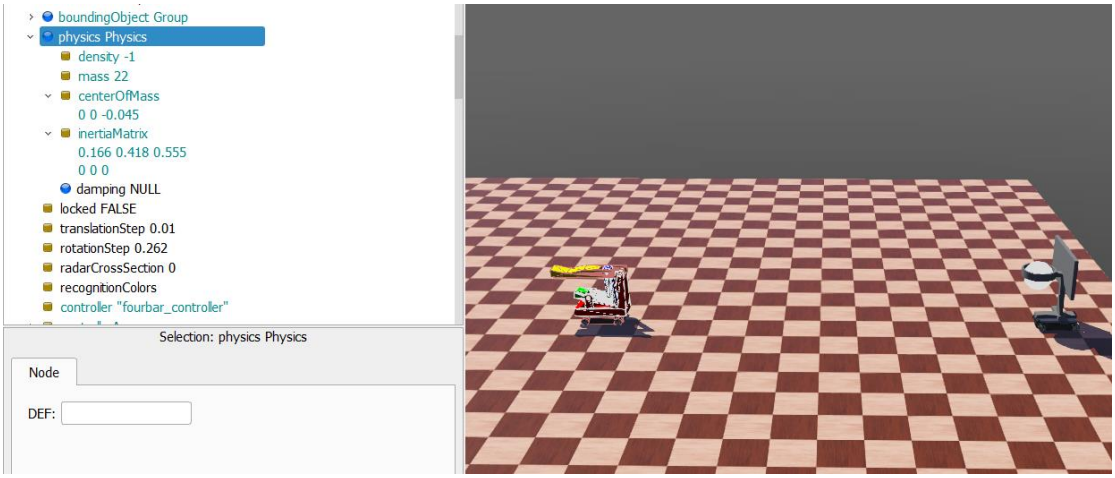

圖 4

在 worldinfo 中設置了兩個 ContactProperties，其中改動的參數有(圖 5)
coulombFriction：滑動摩擦係數
frictionRotation：旋轉摩擦（抗扭轉）
forceDependentSlip：依據力量產生的滑動

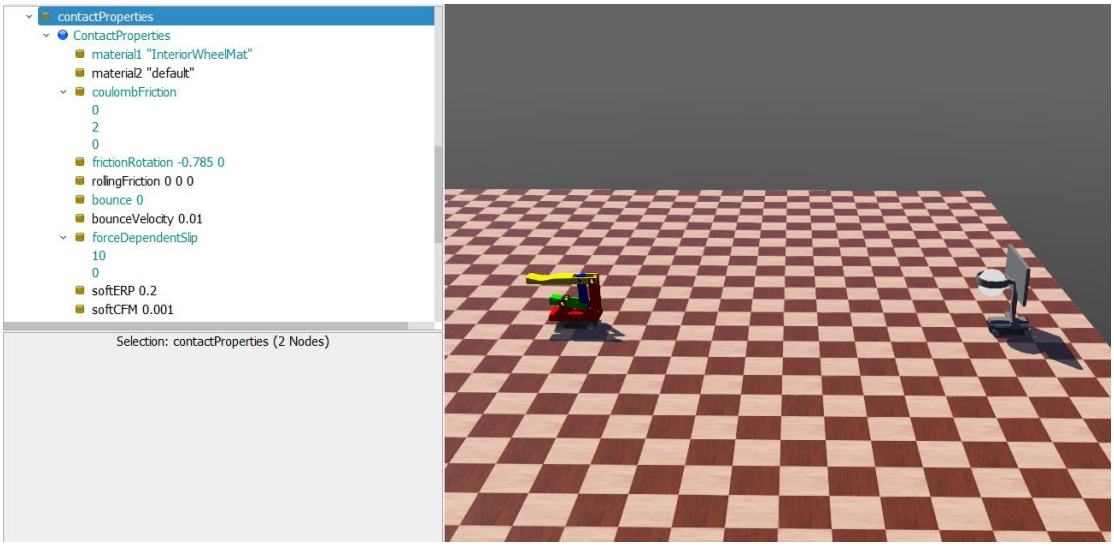

圖 5

2.排錯問題紀錄

1.關於 youbot_stand 到達第一個座標時會卡住的問題，後續找到是因為 InteriorWheel.prote 與 ExteriorWheel.proto 是 webot 本身預設的，雖然他是.proto 檔，但她前身也是一個 robot+hingejoint 群組，將這之下的 motor 的 maxVelocity 預設值 14.8 更改為 100 即可

```
HingeJoint {
    device [
     RotationalMotor {
      name IS name
      maxVelocity 100
     }
```

2.在加入 spervisor 跟 score_board 後，遇到的第一個問題為無法使用 A 鍵生成球，原因是 webot 下的 python 沒有安裝 numpy，只要在 webot 中的 Python command 確認 python 的位址，隨後在使用 python -m pip install numpy 就可以了。

3.接著在 supervisor 下的 feed_ball 又出現問題，由於我自行建立的 youbot 一開始是直接使用 robot 群組下的 name 做更動，但 feed_ball 對於生成球的座標是根據 youbot 改變的，而改變是根據 DEF ... Robot，因此儘管不會顯示錯誤，但仍然無法生成球，因此只要在 feed_ball 下更改我的 DEFname 以及在.wbt 中給 robot 群組加 DEFname 就能做動，內容如下

```
def youbot_local_to_world(local_pos):
    youbot_node = supervisor.getFromDef('youBot_shooter')
    if youbot_node is None:
        raise RuntimeError("找不到 DEF 為 youBot_shooter 的 Robot 物件")
    youbot_translation = np.array(youbot_node.getField('translation').getSFVec3f())
    youbot_rotation = youbot_node.getField('rotation').getSFRotation()
    youbot_axis = youbot_rotation[:3]
    youbot_angle = youbot_rotation[3]
    youbot_rot_mat = axis_angle_to_rotation_matrix(youbot_axis, youbot_angle)
    rotated = youbot_rot_mat @ np.array(local_pos)
    world_pos = youbot_translation + rotated
    return tuple(world_pos)
```

以上設定中就是這一段使生成球的的座標與 shooter 斷開了

```
youbot_node = supervisor.getFromDef('youBot_shooter')
    if youbot_node is None:
        raise RuntimeError("找不到 DEF 為 youBot_shooter 的 Robot 物件")
```

只要給予的 DEFname 相符就能做動

4. 更改 ThreeDigitSevenSegment.proto 的放置情形，使用 score_board supervisor 的 controller 時似乎會無法讀取.proto 的內容，儘管.proto 內容與在.wbt 中使用 Transform 節點的擺設情形是一致的，但這似乎是設定，需要更改成在.wbt 中放置七段顯示器不放置.proto，或是從 controll 下手更改

5. 更改完七段顯示器計分後，隨之而來的問題是無法使用 supervisor 偵測計分，檢查後發現計分上需要用的連動有 emitter 及 receiver，並且他們的 channel 必須一致才能連動，檢測完後開始投籃發現一直無法計分，但似乎是 lookup table 出問題，將他更改並改回來後再把 youbot_shooter 的起始距離往後移動一點即可

而本次內容是更改為 shooter 和 stand 都由玩家操控，使用 B 生成球 J 打擊 K 收回

6. 在自行製作.proto 檔案時發現 proto 檔案無法直接生成，必須得自行建立，尤其是得將"]{" 這兩個括號給分出來分清楚，而以下就是自行建立.proto 檔案的起手內容

之後導入自己的 proto-robot 後 fourbar_controller 無法支援 proto 檔案，根本原因為 proto 檔案會省略掉太多子節點，儘管在 Scite 中檢視也都沒有錯誤，但還是會無法讀取到 motor，接著發現不只因為 proto 檔案無法讀取 motor，導入後將 proto 轉換為 bass node 後也無法讀取，原因是主要 robot 為 youbot 群組，若在下面再加上 shooter 的 robot 群組會發生報錯，所以若想將 shooter 與 youbot 連動，那麼 shooter 必須使用 solid 群組

以下為.proto 檔案建置起手

```
#VRML_SIM R2023b utf8

PROTO shooter1 [
  # 場景控制
  field SFVec3f    translation  0 0 0
  field SFRotation rotation     0 0 1 0
] {
  Robot {
```

(1) Youbot stand node tree

```
DEF youBot_stand Robot {
  translation 6.23 -0.12 0.103
  children [
    BodyMesh {
    }
    DEF WHEEL5 InteriorWheel {
      translation 0.228 -0.158 -0.055
      anchor 0.228 -0.158 -0.055
      name "wheel5"
      sensorName "wheel1sensor"
    }
    DEF WHEEL6 ExteriorWheel {
      translation 0.228 0.158 -0.055
      anchor 0.228 0.158 -0.055
      name "wheel6"
      sensorName "wheel2sensor"
    }
    DEF WHEEL7 ExteriorWheel {
      translation -0.228 -0.158 -0.055
      anchor -0.228 -0.158 -0.055
      name "wheel7"
      sensorName "wheel3sensor"
    }
    DEF WHEEL8 InteriorWheel {
      translation -0.228 0.158 -0.055
      anchor -0.228 0.158 -0.055
      name "wheel8"
      sensorName "wheel4sensor"
    }
    Solid {
      translation 0 0 0.03
      children [
        Pose {
          translation -0.3 0 0.61
          rotation 0 0 1 -1.57
          children [
            CadShape {
```

```
      url [
        "../cad/split_parts/basket_stand_small.obj"
      ]
    }
    Pose {
      children [
        DistanceSensor {
          children [
            DEF sensor_solid Solid {
              children [
                DEF sensor Shape {
                  geometry Sphere {
                    radius 0.2
                  }
                }
              ]
              name "sensor_solid"
            }
          ]
          name "sensor"
          lookupTable [
            0 1000 0.01
            0.12 620 0.01
            0.13 372 0.01
            0.14 248 0.01
            0.15 186 0.01
            0.18 0 0.01
          ]
          type "infra-red"
          numberOfRays 12
        }
      ]
    }
  ]
}
DEF PLATE Shape {
  appearance PBRAppearance {
    baseColor 0.75 0.75 0.75
```

```
        }
        geometry Box {
          size 0.5 0.3 0.02
        }
      }
    ]
    name "PLATE"
    boundingObject USE PLATE
    physics Physics {
      mass 0.5
    }
  }
  GPS {
  }
  InertialUnit {
  }
  Emitter {
    name "score_emitter"
    channel 1
  }
]
name "youBot_stand"
model "KUKA youBot"
description "KUKA youBot - Base with wheels only"
boundingObject Group {
  children [
    Pose {
      translation 0 0 -0.045
      children [
        Box {
          size 0.34 0.34 0.09
        }
      ]
    }
    Pose {
      translation 0 0 -0.045
      children [
        Box {
```

```
            size 0.56 0.23 0.09
          }
        ]
      }
    ]
  }
  physics Physics {
    density -1
    mass 22
    centerOfMass [
      0 0 -0.045
    ]
    inertiaMatrix [
      0.166204 0.418086 0.55459
      0 0 0
    ]
  }
  controller "stand_controller"
}
```

(2) counter_supervisor node tree

```
DEF counter_supervisor Robot {
  children [
    Receiver {
      name "score_receiver"
      channel 1
    }
  ]
  name "counter_supervisor"
  controller "counter_supervisor"
  supervisor TRUE
```

以上(1)(2)是關於第五點提到的 emitter 及 receiver 他們在.wbt 檔案中所在的位置，並且他們的 channel 必須一致才能連動。

(3) stand controller

```
from controller import Robot, Keyboard

# Constants
#TIME_STEP = 32  # Simulation time step in milliseconds
WHEEL_RADIUS = 0.1  # Radius of the wheels in meters (10cm)
L = 0.471  # Half of the robot's length in meters
W = 0.376  # Half of the robot's width in meters
MAX_VELOCITY = 10.0  # Maximum velocity allowed for the wheels

# Initialize the robot
robot = Robot()

# Get simulation time step
timestep = int(robot.getBasicTimeStep())
emitter = robot.getDevice("score_emitter")
score_to_send = 2

# Get the DistanceSensor device
sensor = robot.getDevice('sensor')
sensor.enable(timestep)
score = 0
last_score_time = 0
cooldown = 1.0

# Initialize the keyboard
keyboard = Keyboard()
#keyboard.enable(TIME_STEP)
keyboard.enable(timestep)



# Get motor devices
wheel5 = robot.getDevice("wheel5")  # Front-right wheel
wheel6 = robot.getDevice("wheel6")  # Front-left wheel
wheel7 = robot.getDevice("wheel7")  # Rear-right wheel
wheel8 = robot.getDevice("wheel8")  # Rear-left wheel

# Set motors to velocity control mode
```

```
for wheel in [wheel5, wheel6, wheel7, wheel8]:
    wheel.setPosition(float('inf'))  # Enable velocity control
    wheel.setVelocity(0)  # Set initial velocity to 0

def set_wheel_velocity(v1, v2, v3, v4):
    """Set the velocity of all wheels."""
    wheel5.setVelocity(v1)
    wheel6.setVelocity(v2)
    wheel7.setVelocity(v3)
    wheel8.setVelocity(v4)

# lookupTable 轉成程式用的格式
lookup_table = [
    (1000, 0.00),
    (620, 0.12),
    (372, 0.13),
    (248, 0.14),
    (186, 0.15),
    (0, 0.18)
]

def ad_to_distance(ad_value):
    # 假設 AD 值遞減，距離遞增
    for i in range(len(lookup_table)-1):
        a0, d0 = lookup_table[i]
        a1, d1 = lookup_table[i+1]
        if a1 <= ad_value <= a0:
            # 線性插值
            return d0 + (d1 - d0) * (ad_value - a0) / (a1 - a0)
    # 超出範圍時回傳極值
    if ad_value > lookup_table[0][0]:
        return lookup_table[0][1]
    return lookup_table[-1][1]

# Main loop
print("Use 'W', 'A', 'S', 'D' keys to control the robot.")
print("W: Move forward, S: Move backward, A: Turn left, D: Turn
right.")
```

```
print("Press 'Q' to quit.")

#while robot.step(TIME_STEP) != -1:
while robot.step(timestep) != -1:

    key = keyboard.getKey()  # Read the key pressed
    # Read DistanceSensor value
    sensor_value = sensor.getValue()
    #print(sensor_value)
    distance = ad_to_distance(sensor_value)
    current_time = robot.getTime()
    #print(sensor_value)
    # Check if the ball blocks the sensor (you may need to adjust the
threshold based on your sensor's range)
    if key == ord('J') or key == ord('j'):
        print(distance)
    if key == ord('K') or key == ord('k'):
        print(distance)

    if distance < 0.11 and (current_time - last_score_time) >
cooldown:
        score +=2
        print("得分")
        print(distance)
        emitter.send(str(score_to_send))



    if key == ord('S') or key == ord('s'):
        # Move forward
        velocity = MAX_VELOCITY
        set_wheel_velocity(velocity, velocity, velocity, velocity)
    elif key == ord('W') or key == ord('w'):
        # Move backward
        velocity = -MAX_VELOCITY
        set_wheel_velocity(velocity, velocity, velocity, velocity)
    elif key == ord('D') or key == ord('d'):
        # Turn right
```

```
        velocity = MAX_VELOCITY
        set_wheel_velocity(-velocity, velocity, -velocity, velocity)
    elif key == ord('A') or key == ord('a'):
        # Turn left
        velocity = MAX_VELOCITY
        set_wheel_velocity(velocity, -velocity, velocity, -velocity)
    elif key == ord('Q') or key == ord('q'):
        # Quit the program
        print("Exiting...")
        break
    else:
        # Stop the wheels when no key is pressed
        set_wheel_velocity(0, 0, 0, 0)
```

(4) counter controller

```
from controller import Supervisor

SEGMENTS = [
    [1,1,1,1,1,1,0], # 0
    [0,1,1,0,0,0,0], # 1
    [1,1,0,1,1,0,1], # 2
    [1,1,1,1,0,0,1], # 3
    [0,1,1,0,0,1,1], # 4
    [1,0,1,1,0,1,1], # 5
    [1,0,1,1,1,1,1], # 6
    [1,1,1,0,0,0,0], # 7
    [1,1,1,1,1,1,1], # 8
    [1,1,1,1,0,1,1], # 9
]
DIGIT_MATERIALS = [
    ['a3mat', 'b3mat', 'c3mat', 'd3mat', 'e3mat', 'f3mat', 'g3mat'],
# 百
    ['a2mat', 'b2mat', 'c2mat', 'd2mat', 'e2mat', 'f2mat', 'g2mat'],
# 十
    ['a1mat', 'b1mat', 'c1mat', 'd1mat', 'e1mat', 'f1mat', 'g1mat'],
# 個
]
ON_COLOR = [0, 1, 0]
OFF_COLOR = [0.05, 0.05, 0.05]

def set_digit(supervisor, digit_index, value):
    segs = SEGMENTS[value]
    for i, seg_on in enumerate(segs):
        mat_node =
supervisor.getFromDef(DIGIT_MATERIALS[digit_index][i])
        if mat_node:
            mat_node.getField('diffuseColor').setSFColor(ON_COLOR if
seg_on else OFF_COLOR)
        else:
            print(f"找不到 {DIGIT_MATERIALS[digit_index][i]} 這個
DEF")
```

```
def set_display(supervisor, value):
    value = max(0, min(999, int(value)))
    h = value // 100
    t = (value // 10) % 10
    u = value % 10
    set_digit(supervisor, 0, h)
    set_digit(supervisor, 1, t)
    set_digit(supervisor, 2, u)

supervisor = Supervisor()
timestep = int(supervisor.getBasicTimeStep())

score = 0
receiver = supervisor.getDevice("score_receiver")
receiver.enable(timestep)

while supervisor.step(timestep) != -1:
    while receiver.getQueueLength() > 0:
        data = receiver.getString()
        if data.isdigit():
            try:
                received_score = int(data)
                score += received_score
                print(f"收到得分訊息: +{received_score}, 總分:
{score}")
            except Exception as e:
                print("訊息格式錯誤:", e)
        receiver.nextPacket()
    set_display(supervisor, score)
```

以上是關於第五點 counter 以及 stand 的連動原因，他們主要由其中的

```
timestep = int(robot.getBasicTimeStep())
emitter = robot.getDevice("score_emitter")
score_to_send = 2


if distance < 0.11 and (current_time - last_score_time) > cooldown:
      score +=2
```

```
        print("得分")
        print(distance)
        emitter.send(str(score_to_send))
```

以及

```
while supervisor.step(timestep) != -1:
    while receiver.getQueueLength() > 0:
        data = receiver.getString()
        if data.isdigit():
            try:
                received_score = int(data)
                score += received_score
                print(f"收到得分訊息: +{received_score},  總分: {score}")
            except Exception as e:
                print("訊息格式錯誤:", e)
        receiver.nextPacket()
    set_display(supervisor, score)
```

來傳送訊息