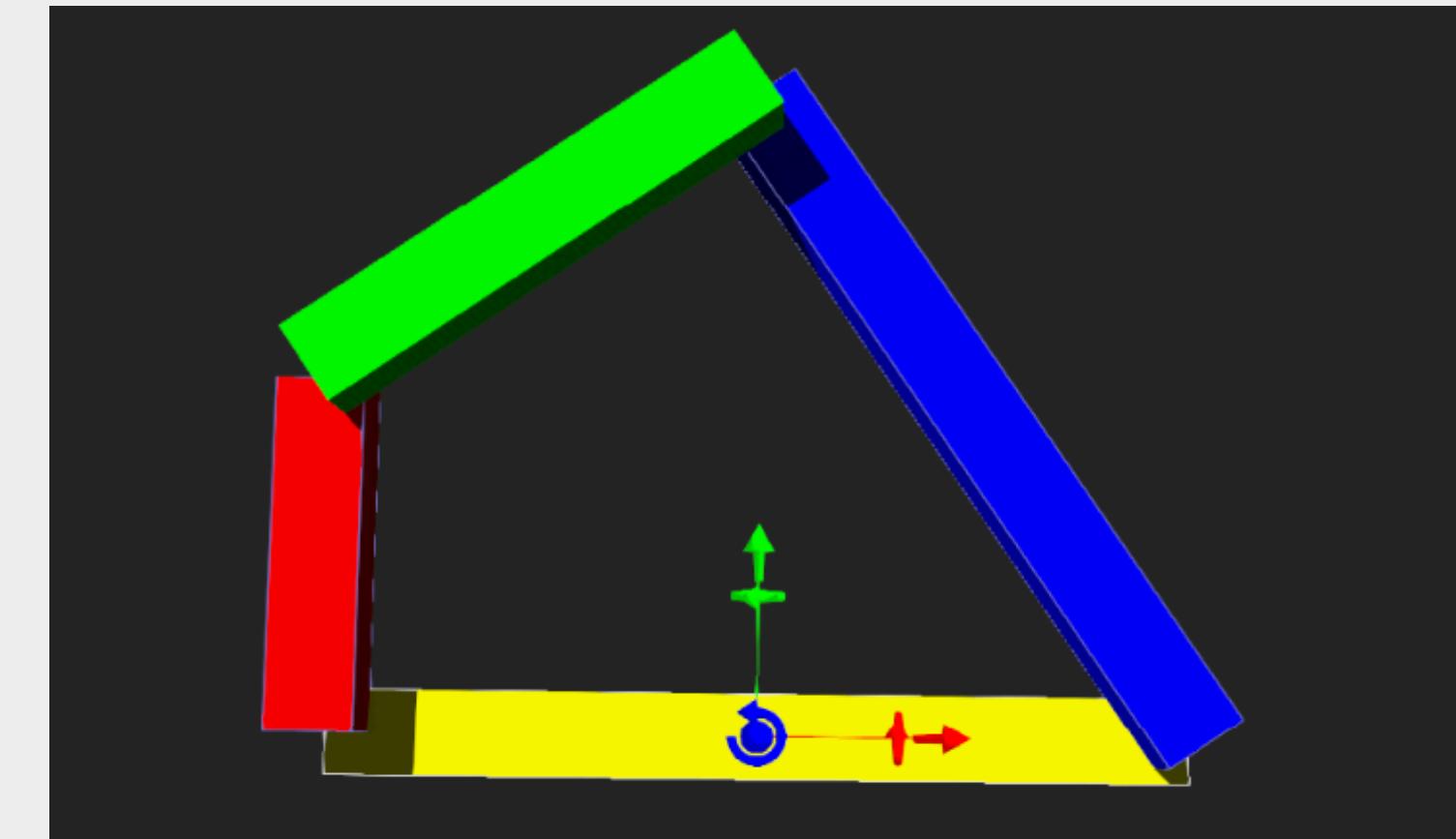


協同產品設計實習

41223110	丁尉倫
41223125	林家宏
41223129	洪義凱
41223132	莊宏恩
41223147	蔡福璟

HW1

新增robot，先建立base基準桿，新增Shape設定好長度1m與顏色後，在跟base同階位的地方建立兩個Higejoint，為joint1與joint4。在joint1下增加Solid設定joint1的旋轉軸參數與座標，新增rotational motor，接著新增Pose→Shape設定link1的尺寸0.4m與顏色。在跟link1_Pose同階位的地方新增Higejoint joint2，設定joint2的旋轉軸參數與座標後新增Pose→Shape設定link2的尺寸0.6m與顏色。在跟link2_Pose同階位的地方新增Higejoint joint3，設定joint3的旋轉軸參數與座標後新增Pose→Shape設定link3的尺寸0.9m與顏色。最後就是joint4，設定好joint4的旋轉軸參數與座標後選擇跟隨link3。檢查每個碰撞模型(boundingObject)都有選對link後，物理性質也有開起就可以匯入python程式，這樣就完成了。



python模擬程式:

```
1  from controller import Robot  
2  
3  def run_robot():  
4      # Create the Robot instance  
5      robot = Robot()  
6  
7      # Get simulation time step  
8      timestep = int(robot.getBasicTimeStep())  
9  
10     # Get motor device  
11     motor = robot.getDevice('motor')  
12  
13     # Set motor for continuous rotation  
14     motor.setPosition(float('inf'))  
15     motor.setVelocity(1.0)  
16  
17     # Main control loop  
18     while robot.step(timestep) != -1:  
19         pass  
20  
21     if __name__ == "__main__":  
22         run_robot()
```

HW2

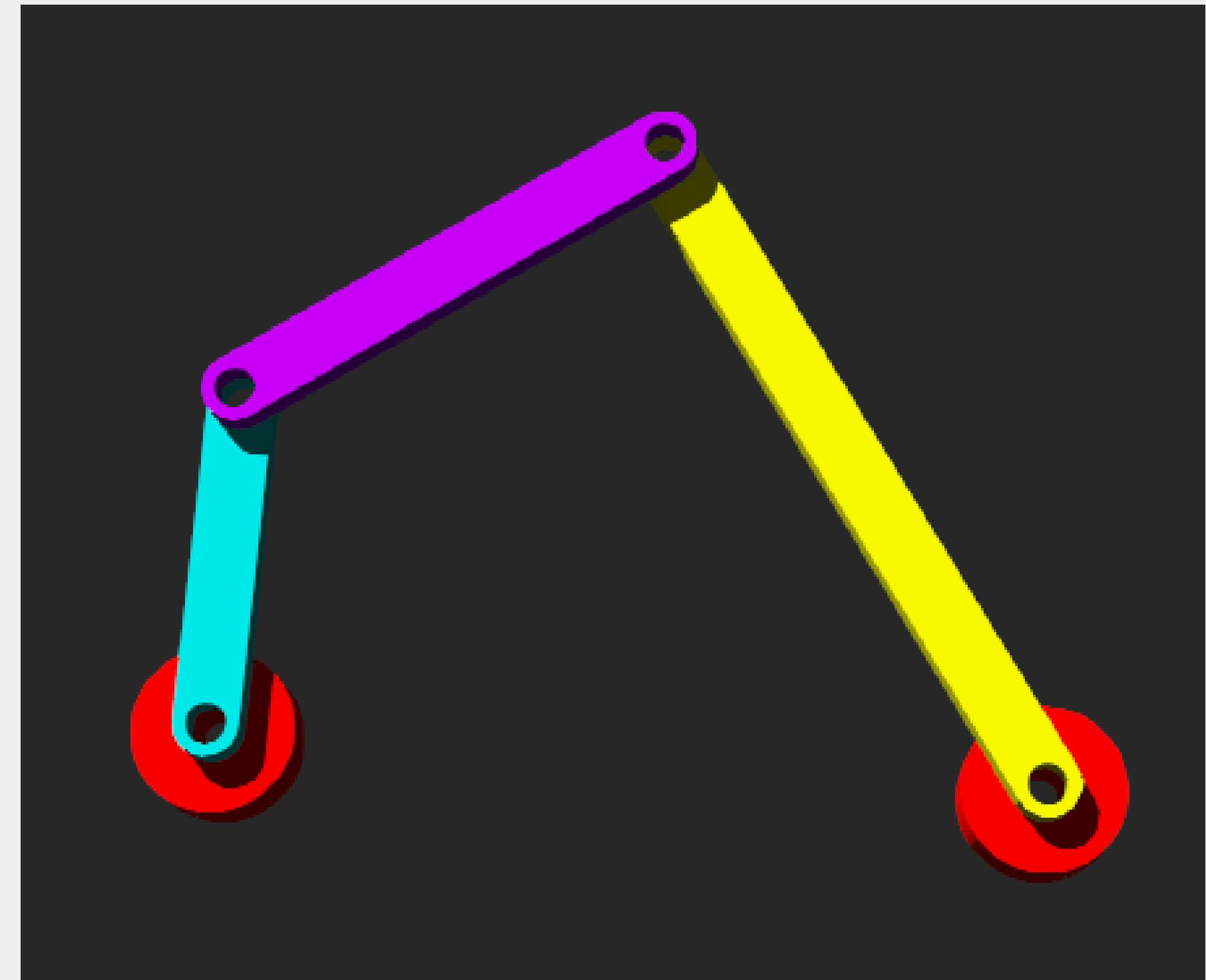
要先在solvespace畫好每個零件，HW2 link1、link2、link3的長度都是跟HW1一樣的，組裝起來後轉成stl檔，接著最重要的一步是要把stl檔轉成obj檔，這樣才可以在webots裡面匯入零件。

建立Robot，新增兩個Solid與一個Higejoint，兩個solid為兩邊的基座，Higejoint為link1的joint1。

在base1 Solid下新增Shape，新增Mash後選擇base1的obj檔並在diffuseColor設定顏色，base2 Solid也是同樣的步驟。

在Higejoint joint1下先設定旋轉軸參數，新增rotational motor，新增link1 Solid→Shape新增Mash後選擇link1的obj檔並在diffuseColor設定顏色，在跟link1_Shape同階位的地方新增Higejoint joint2，新增link2 Solid→Shape新增Mash後選擇link2的obj檔並在diffuseColor設定顏色，在跟link2_Shape同階位的地方新增Higejoint joint3，新增link3 Solid→Shape新增Mash後選擇link3的obj檔並在diffuseColor設定顏色。

檢查每個碰撞模型(boundingObject)都有選對跟隨的obj檔後，物理性質也有開起就可以匯入python程式，這樣就完成了。



HW2

python模擬程式:

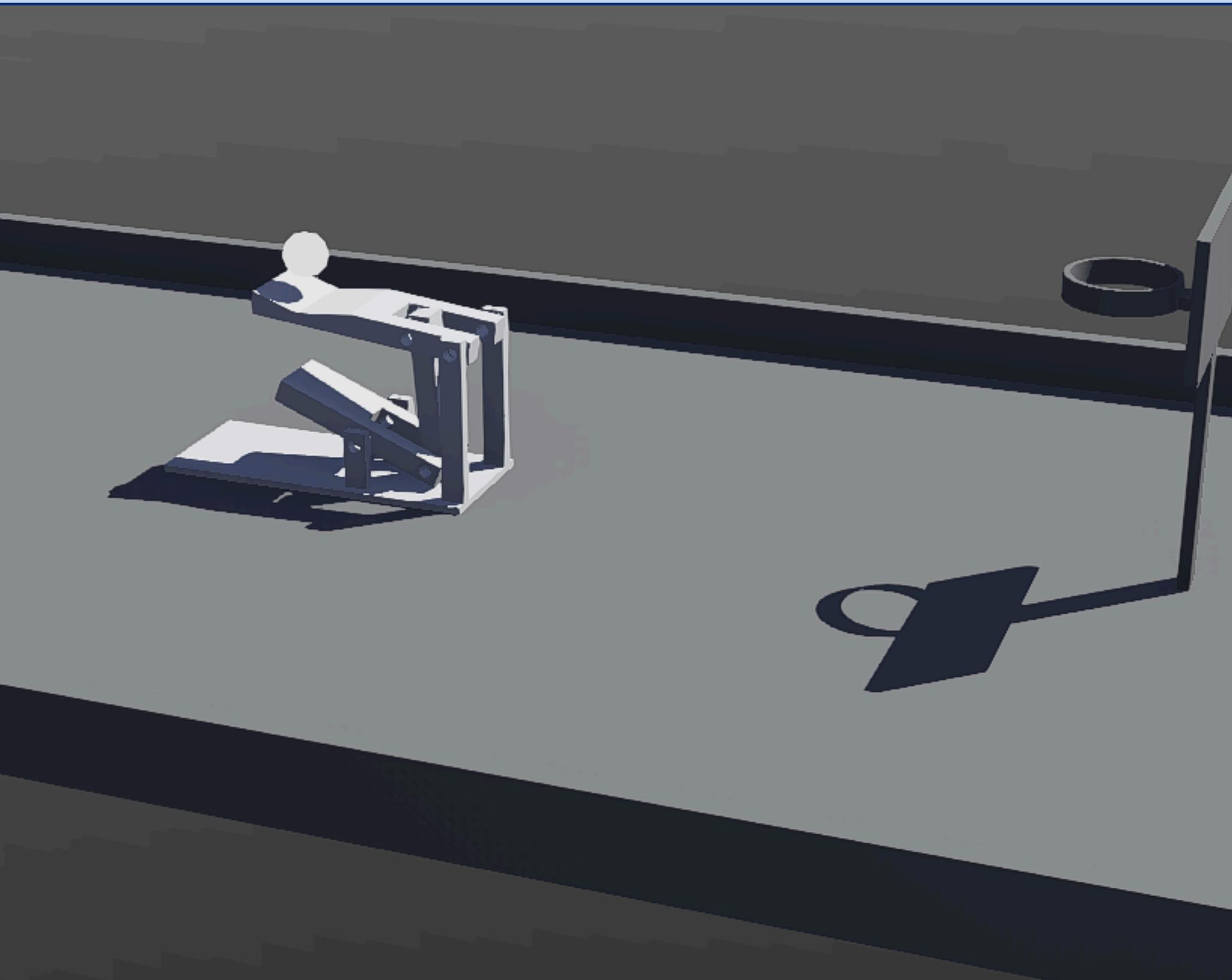
```
1  from controller import Robot  
2  
3  def run_robot():  
4      # Create the Robot instance  
5      robot = Robot()  
6  
7      # Get simulation time step  
8      timestep = int(robot.getBasicTimeStep())  
9  
10     # Get motor device  
11     motor = robot.getDevice('motor')  
12  
13     # Set motor for continuous rotation  
14     motor.setPosition(float('inf'))  
15     motor.setVelocity(1.0)  
16  
17     # Main control loop  
18     while robot.step(timestep) != -1:  
19         pass  
20  
21     if __name__ == "__main__":  
22         run_robot()
```

stl轉obj程式

```
# Write the corresponding MTL file  
self._write_mtl(mtl_filename, material_name)  
  
def split_and_convert(self):  
    """分割 STL 檔案並轉換為 OBJ 格式"""  
    if self.is_binary:  
        triangles, normals = self._read_binary_stl()  
    else:  
        triangles, normals = self._read_ascii_stl()  
  
    components = self._split_by_connected_components(triangles, normals)  
  
    output_dir = Path('456')  
    output_dir.mkdir(parents=True, exist_ok=True)  
  
    for i, component in enumerate(components):  
        component_triangles = triangles[component]  
        component_normals = normals[component]  
  
        base_name = f"part_{i + 1}"  
        stl_filename = output_dir / f"{base_name}.stl"  
        obj_filename = output_dir / f"{base_name}.obj"  
  
        #self._write_binary_stl(stl_filename, component_triangles, component_normals)  
        self._write_obj(obj_filename, component_triangles, component_normals)  
  
        print(f"已儲存零件 {i + 1} 到:")  
        #print(f' STL: {stl_filename}')  
        print(f' OBJ: {obj_filename}')  
        print(f' MTL: {obj_filename.with_suffix('.mtl')}")  
  
    return len(components)  
  
# 直接轉換指定的 STL 檔案  
try:  
    # 指定要轉換的 STL 檔案名稱和縮放比例  
    stl_file = "exam2_assemble.stl" ← 此為stl檔名  
    scale = 0.01 # 縮放比例, 可以根據需要調整  
  
    # 創建轉換器實例並執行轉換  
    converter = STLConverter(stl_file, scale=scale)  
    num_parts = converter.split_and_convert()  
    print(f"\n總共處理了 {num_parts} 個零件")  
except Exception as e:  
    print(f"錯誤: {e}")
```

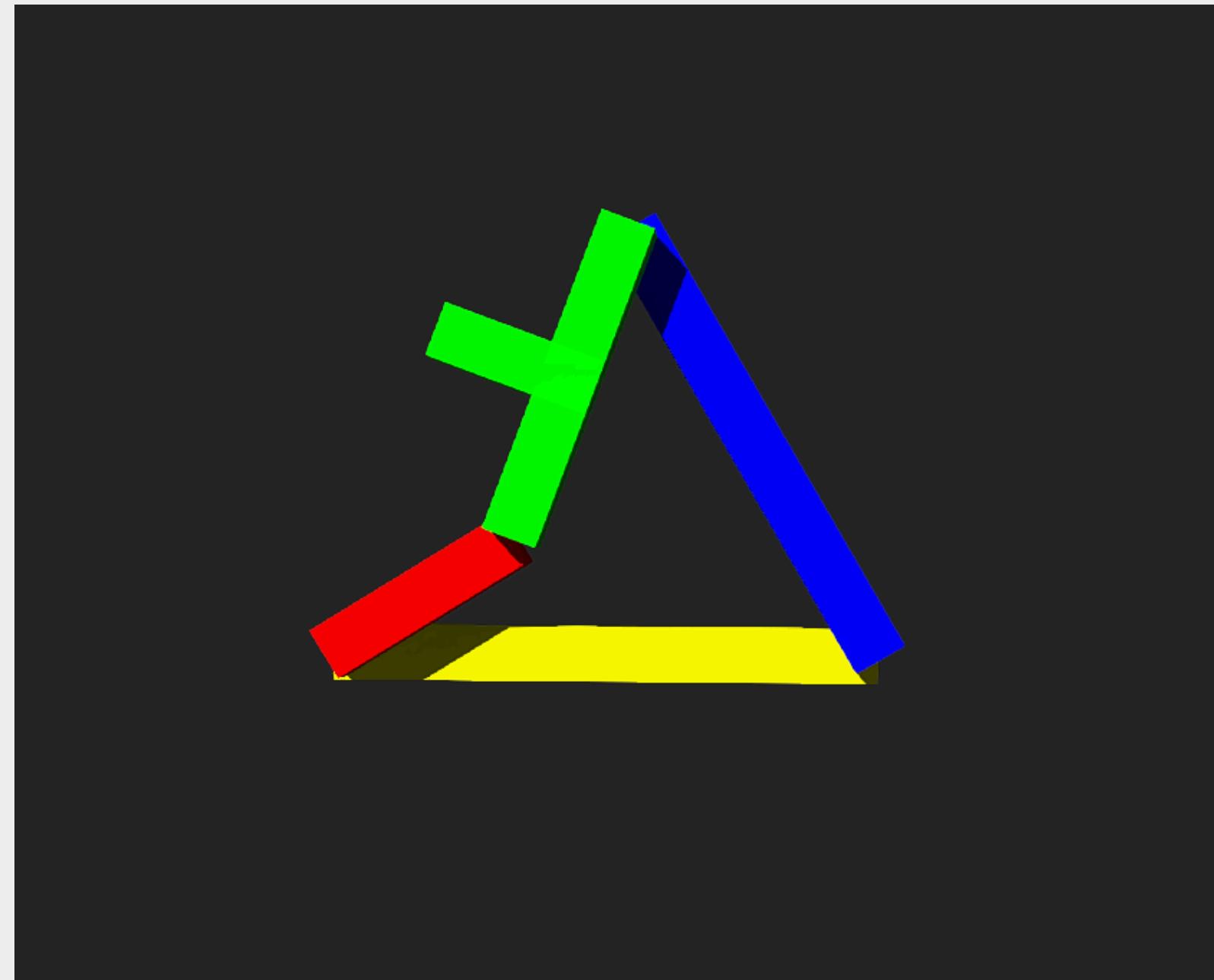
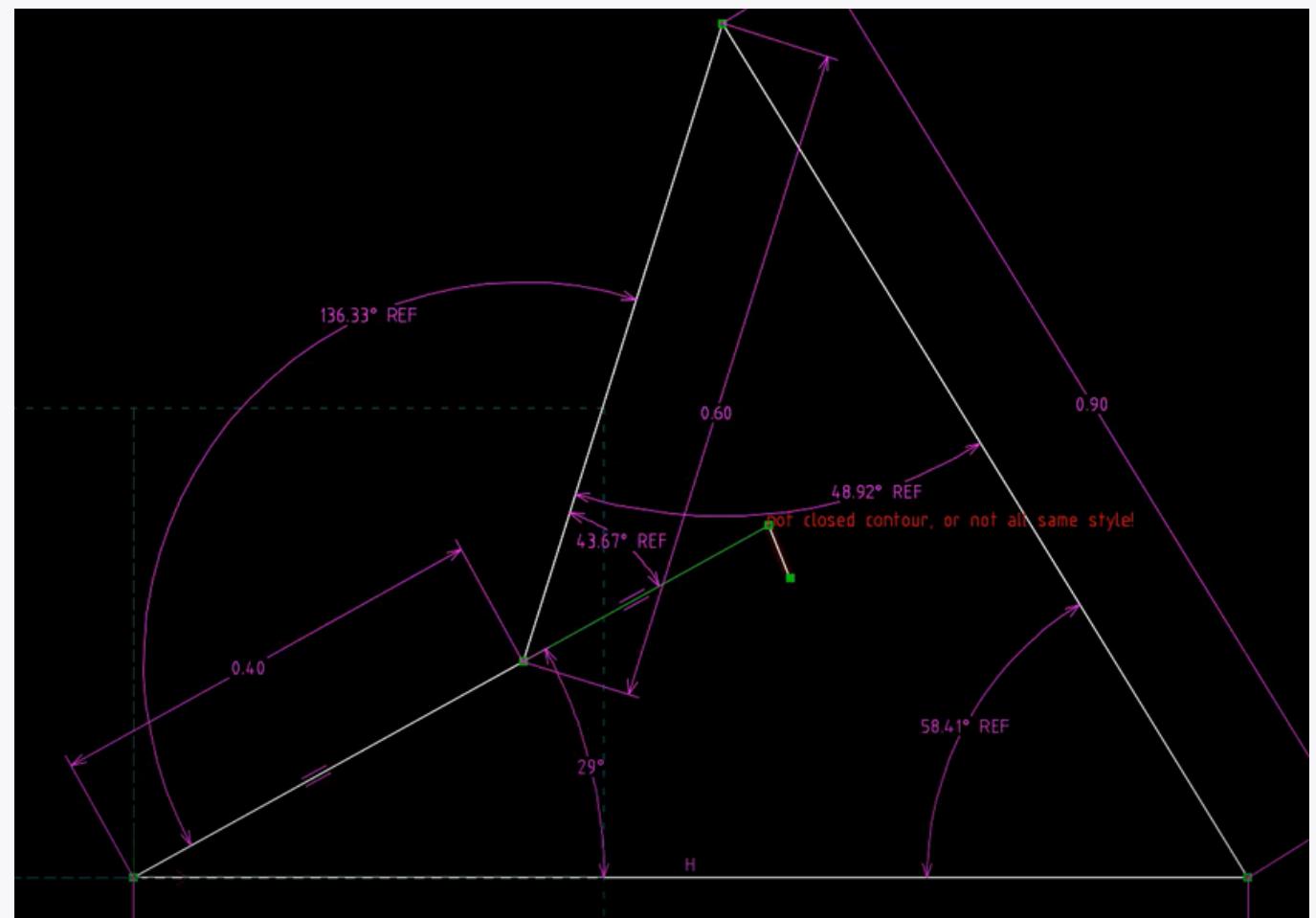
HW3

HW3跟HW2一樣要先在 solvespacec 繪製好零件並組合好轉stl檔再轉obj檔，開啟 webots 建立Robot，在Robot 下加入 solid(base)和兩個 hingejoint，剩下的做法 與 HW2相似加入零件與修改內部 數值，不一樣的地方在於需要 多加ball與籃球框和地板，設 定球的滾動方向及調整motor 的力量，最後匯入pythont程式



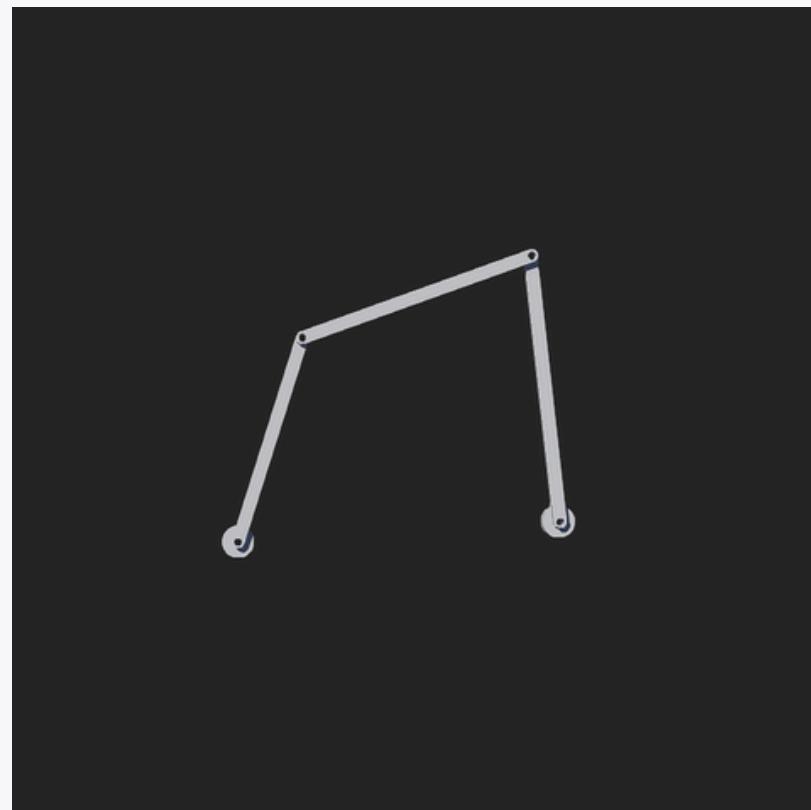
Exam1

Exam1在原有的HW1的基礎上需新增一個長度為 0.3m的短桿，紅色連桿link1旋轉角度需為學號的後兩碼

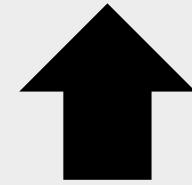


Exam2

Ex2的部分也是先繪圖後轉檔，與Hw2做法類似，輸入端點位置慢慢連接連桿，在設定一個馬達使短桿以轉速 1 rad/s 旋轉。

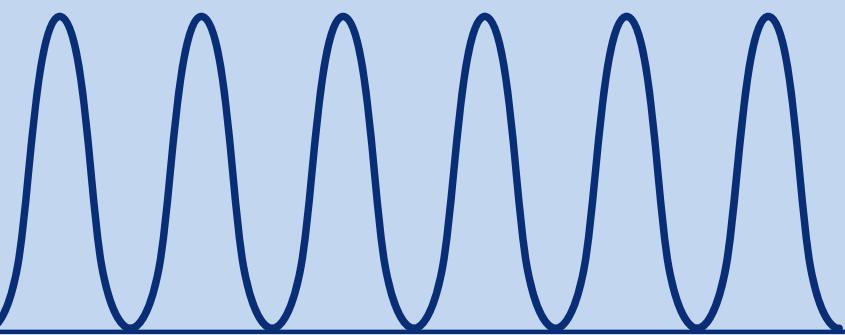


```
DEF robot DEFEXT  
  rotation 0 0 1 0  
  children  
    DEF base1 Solid  
    DEF base2 Solid  
    DEF joint1 HingeJoint  
      jointParameters HingeJointParameters  
      device  
        RotationalMotor "motor"  
    endPoint Solid  
    DEF joint4 HingeJoint  
  name "robot"  
  model ""  
  description ""  
  contactMaterial "default"
```



在短桿處加入馬達使其運轉，兩個base是要固定兩個輪子。

Exam-3-1 webotsw --stream



The screenshot shows the 'Webots Streaming Viewer' application window. On the left, there's a 3D rendering of a robot arm with a green and red segment. On the right, a terminal-like interface displays the log output of the Python script 'Exam3-HW1-server.py'. The log shows various messages from the Webots simulation, including 'Simulation: 重設 Webots 世界 (simulationReset) 並馬達停止' (Simulation: Reset Webots world (simulationReset) and motor stop), 'Error: ignoring illegal call to wb_supervisor_simulation_reset() in Error: this function can only be used in a 'Supervisor' controller.', and repeated messages about receiving control commands via WebSocket.

```
Exam3-HW1-server.py - SciTE
File Edit Search View Tools Options Language Buffers Help
1 Exam3-HW1-server.py
resume_simulation()
time.sleep(0.5)
time.sleep(0.05)
except Exception as e:
    print(f"Error in arrow key control: {e}")
    break

def start_webots_simulation():
    try:
        while robot.step(TIME_STEP) != -1:
            key = webots_keyboard.getKey()
            if key == 27: # ESC
                print("[反饋] Webots鍵盤: ESC -> 暫停 simulation 並馬達停止")
                pause_simulation()
            elif key in (ord('q'), ord('Q')):
                print("[反饋] Webots鍵盤: q (馬達啟動)")
                set_wheel_velocity(MAX_VELOCITY)
            elif key in (ord('r'), ord('R')):
                print("[反饋] Webots鍵盤: r -> 重設 simulation 並馬達停止")
                reset_simulation()
            elif key in (ord('e'), ord('E')):
                print("[反饋] Webots鍵盤: e -> 復復 simulation")
                resume_simulation()
            except Exception as e:
                print(f"Simulation error: {e}")

    if __name__ == "__main__":
        websocket_thread = threading.Thread(target=start_websocket_server, daemon=True)
        simulation_thread = threading.Thread(target=start_webots_simulation, daemon=True)
        arrow_key_thread = threading.Thread(target=start_arrow_key_control, daemon=True)

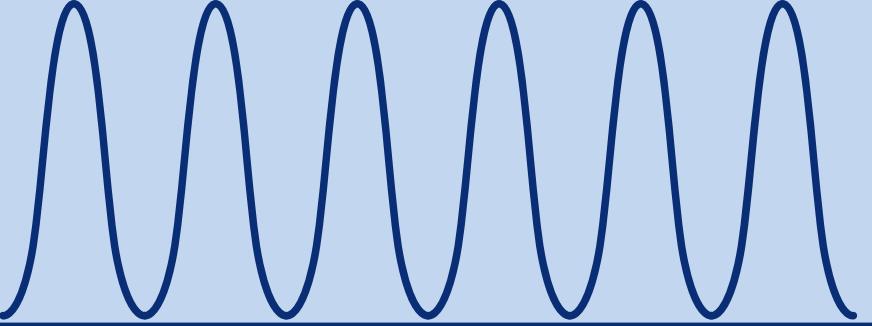
        websocket_thread.start()
        simulation_thread.start()
        arrow_key_thread.start()

    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        print("Shutting down gracefully...")
```

```
[反饋] Simulation: 重設 Webots 世界 (simulationReset) 並馬達停止
Error: ignoring illegal call to wb_supervisor_simulation_reset() in
Error: this function can only be used in a 'Supervisor' controller.
收到控制訊息: {"direction": "q"}
[反饋] WebSocket: 收到 q, 馬達啟動
收到控制訊息: {"direction": "q"}  
[反饋] Simulation: 重設 Webots 世界 (simulationReset) 並馬達停止
Error: ignoring illegal call to wb_supervisor_simulation_reset() in
Error: this function can only be used in a 'Supervisor' controller.
[反饋] 本地鍵盤: ESC -> 暫停 simulation 並馬達停止
[反饋] Simulation: 重設 Webots 世界 (simulationSetMode PAUSE) 並馬達停止
Error: ignoring illegal call to wb_supervisor_simulation_set_mode() in
Error: this function can only be used in a 'Supervisor' controller.
[反饋] 本地鍵盤: r -> 重設 simulation 並馬達停止
[反饋] Simulation: 重設 Webots 世界 (simulationReset) 並馬達停止
Error: ignoring illegal call to wb_supervisor_simulation_reset() in
Error: this function can only be used in a 'Supervisor' controller.
[反饋] 本地鍵盤: r -> 重設 simulation 並馬達停止
[反饋] Simulation: 重設 Webots 世界 (simulationReset) 並馬達停止
Error: ignoring illegal call to wb_supervisor_simulation_reset() in
Error: this function can only be used in a 'Supervisor' controller.
[反饋] 本地鍵盤: q (馬達啟動)
[反饋] 本地鍵盤: q
[反饋] 本地鍵盤: q
[反饋] 本地鍵盤: q
[反饋] 本地鍵盤: q
```

Webots 控制器 Python 程式碼，整合WebSocket 通訊、鍵盤輸入（本地與 Webots 鍵盤）、Supervisor 控制功能與多執行緒設計，主要用來遠端或本地控制 Webots 模擬中馬達的行為，以及控制模擬器的狀態（重設、暫停、啟動）。

Exam-3-2 七段顯示器



```
1 Exam3-2_server_ag6.py  2 Exam3-2_client_ag6.py
# pip install websocket-client keyboard
import websocket
import json
import keyboard
import time

def send_command(cmd):
    server_ip = "120.113.99.46" # 改成你的 IPv4 位址
    server_port = 8081
    ws_url = f"ws://({server_ip}):({server_port})" # 不需要中括號，僅限 IPv6

    try:
        ws = websocket.create_connection(ws_url)
        message = json.dumps({"cmd": cmd})
        ws.send(message)
        result = ws.recv()
        print(f"Sent command: {cmd} | Received: {result}")
        ws.close()
    except Exception as e:
        print(f"Failed to send command {cmd}: {e}")

def main():
    print("按 W (上一面) , S (下一面) , ESC 離開。")
    while True:
        try:
            if keyboard.is_pressed("w"):
                send_command("w")
            while keyboard.is_pressed("w"):
                time.sleep(0.1) # 等待放開避免重複送
            elif keyboard.is_pressed("s"):
                send_command("s")
            while keyboard.is_pressed("s"):
                time.sleep(0.1)
            elif keyboard.is_pressed("esc"):
                print("Exiting...")
                break
        except Exception as e:
            print(f"Error: {e}")
            break

if __name__ == "__main__":
    main()
```

```
>pythonw.exe -u "Exam3-2_client_ag6.py"
按 W (上一面) , S (下一面) , ESC 離開。
wSent command: W | Received: W切换到: 29
sSent command: S | Received: S切换到: 47
sSent command: S | Received: S切换到: 25
sSent command: S | Received: S切换到: 32
wSent command: W | Received: W切换到: 47
wSent command: W | Received: W切换到: 10
```

seven 程式:

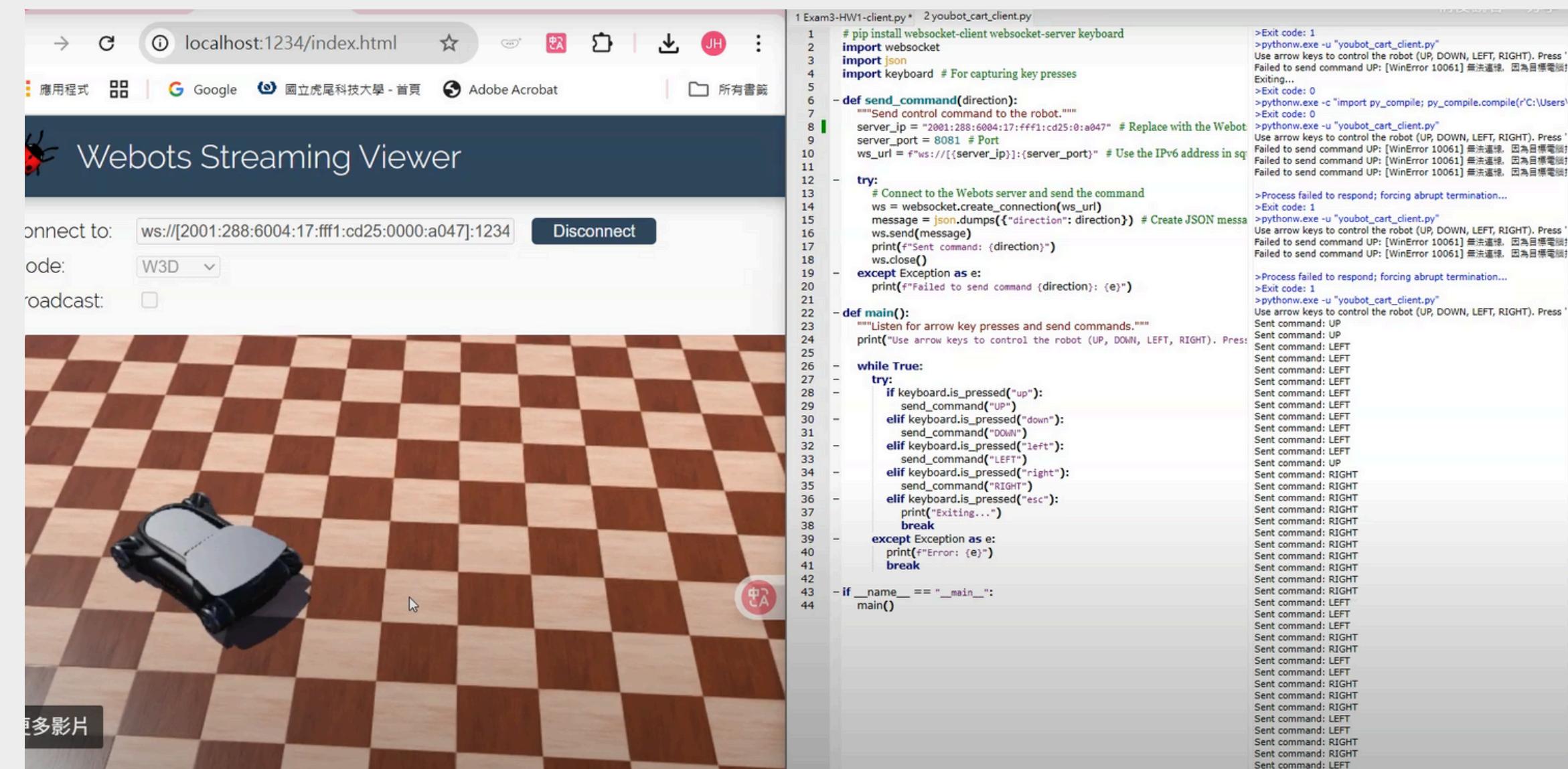
- 初始化七段顯示器的每個 segment 材質節點：
- 利用 getFromDef() 取得 Webots 中每個 segment 的 DEF 名稱對應的 node。
- 利用 getField("diffuseColor") 取得該 segment 的顏色控制欄位。
- 定義了每個阿拉伯數字（0-9）對應的七段開關模式（共7段）。
- 透過 display_number() 方法，把整個三位數字（百位、十位、個位）依序設定上去。
- 在 simulation loop 裡持續顯示數字（目前為固定值123）

控制顏色變數

變數名稱	說明	範例值
self.color_on	七段 segment 亮起的顏色	[0.0, 1.0, 0.0] (亮綠)
self.color_off	七段 segment 熄滅時的顏色	[0.0, 0.0, 0.0] (黑色)

修 改 範 圍 值 得 漫 反 射 顏 色
(diffuseColor
來改變color_on及color_off的顏色

Exam-3-3 平台車



具備：

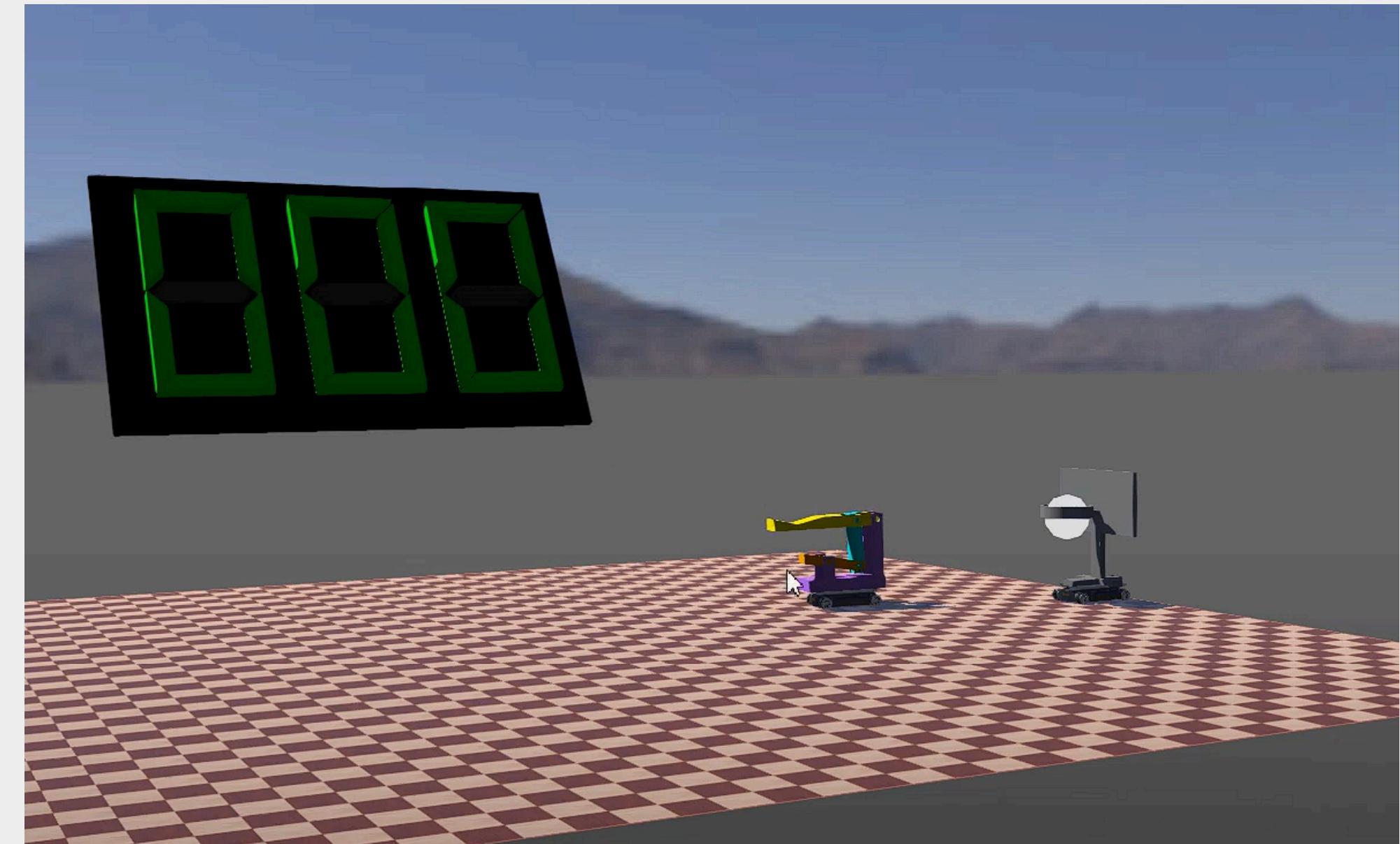
- 遠端控制（WebSocket）
- 本地鍵盤控制
- 多執行緒架構
- 可擴充感測器、GUI 等功能

WebSocket 控制從瀏覽器或其他程式傳送 JSON 指令，例如 "UP"、"LEFT" 來控制機器人，本地鍵盤控制在終端機按鍵（上、下、左、右）直接控制機器人，Webots 模擬器啟動 Webots simulation 並持續步進（可擴充為感測器等邏輯）。

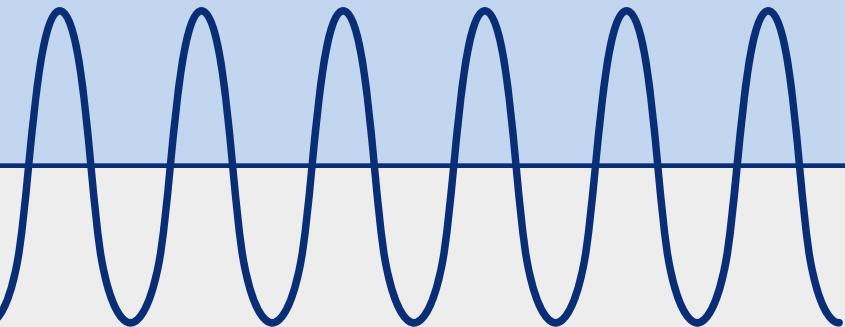
Final

將籃球機與籃框放置在平台車上，讓平台車載著籃球機移動投球，投進後分數板會增加分數。這個專案對我們來說很困難時，一旦有出現一個錯誤的設定的地方或是沒有設定到的地方就沒辦法進行模擬，因為很多地方都是有前後關係的。

我們在做的過程不斷的詢問GPT程式，像是平台車的移動、補球數量、記分板、把球投出與籃框的感應等等，我們把程式修改完整能操控平台車與投球機。



學習成果



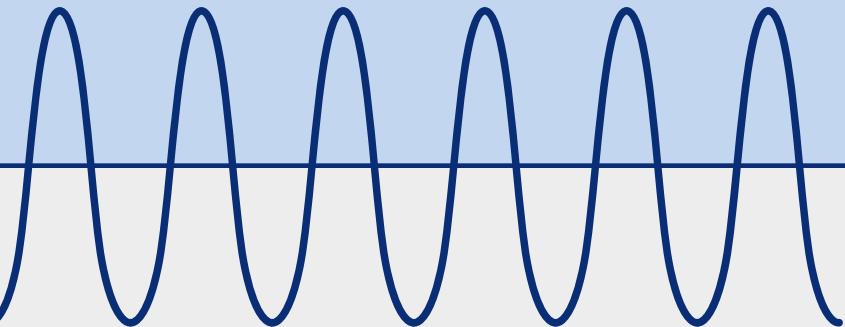
- Tutorial6 的主要學習主題是什麼？
- Tutorial7 的核心任務是什麼？

從頭開始創建一個機器人
在 Webots 中創建的任何節點創建
PROTO 節點。

- Tutorial8 的主要學習目標是什麼？

就自行開發投籃機模擬場景而言，可在
虛擬的模擬場景中，利用 supervisor 模
式配置非可實際達成的程式控制加速虛
實產品的開發流程。

學習成果



- 場景中為何有兩個 BASE (BASE 與 BASE2) ? 固定兩個輪子
- 為什麼每個連桿 (link) 都需要 physics 與 boundingObject ? 參與碰撞與模擬、動力學模擬
- HingeJoint 節點在本結構中如何應用 ? 建立一維旋轉自由度的關節