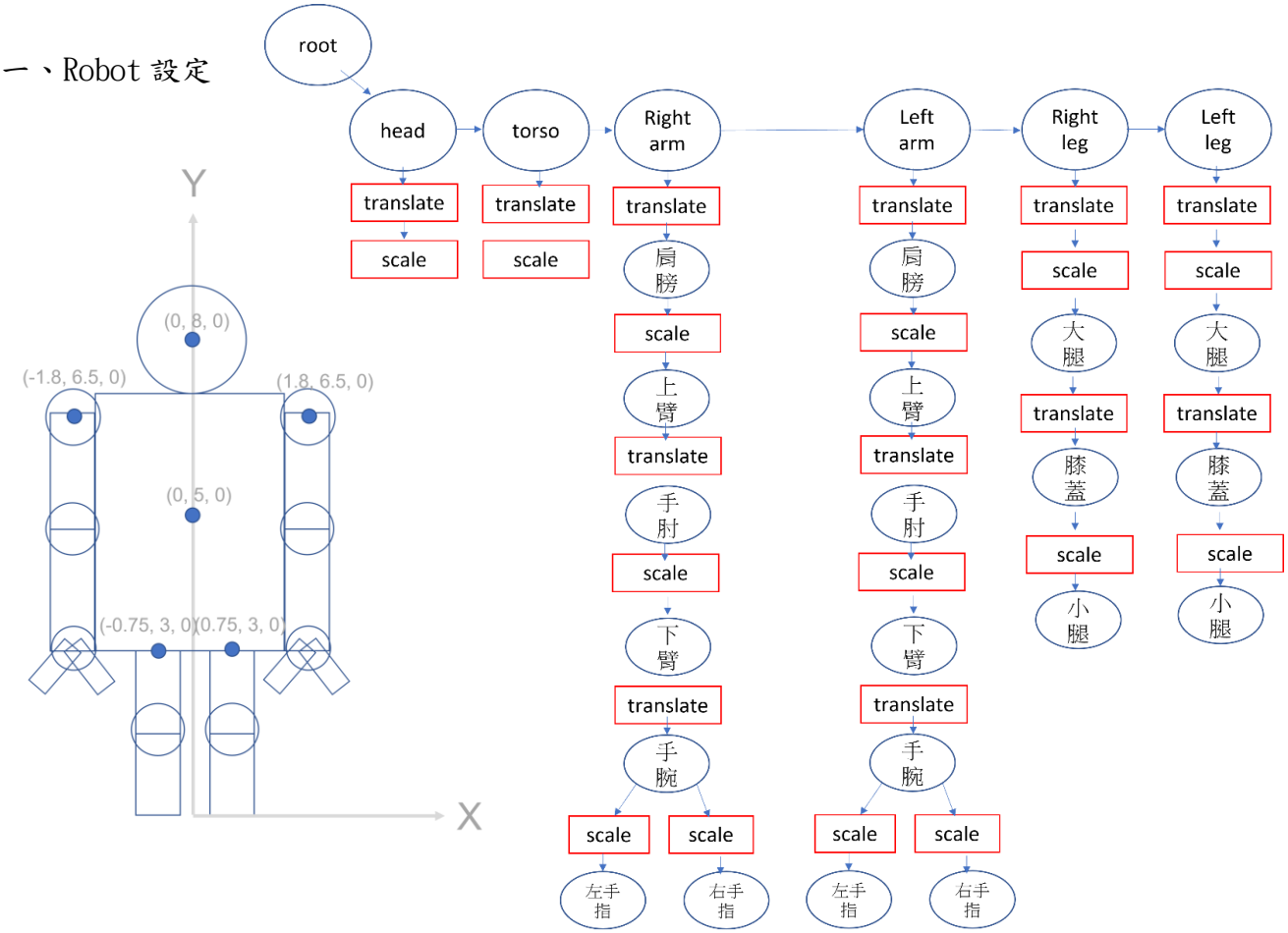
Computer Graphics Project #2, A moving robot

01157006 陳柏秀

一、Robot 設定



關節名稱	自由度與描述	角度
肩膀	X 軸和 Z 軸旋轉	X 軸：-45° ~ +45° Z 軸：-90° ~ +90°
手肘	X 軸旋轉（前後彎曲動作）	0° ~ +135°
手腕	X 軸和 Y 軸旋轉	X 軸：-30° ~ +30° Y 軸：-45° ~ +45°
膝蓋	X 軸旋轉（只能前後彎曲）	0° ~ -120°
大腿	X 軸旋轉（步行時腳擺動）	走路 ： -40° ~ +40° 跑步 ： -70° ~ +70°

二、操作說明

鍵盤	動作
w	向前走
Shiht + w	向前跑
s	向後走
Shift + s	向後跑
a	左轉
d	右轉
g	抓握
j	跳躍
k	蹲下
l	揮手
z	切換視角跟隨模式
x	舞蹈
Space	發射子彈

三、實現方法

- 走路及跑步
  - 設定步伐增量：根據行走模式設定步伐大小，即每次前進或後退的距離。
    步行模式：設定較小的步伐增量，例如 step = 0.2，使機器人以較慢的速度移動。
    跑步模式：設定較大的步伐增量，例如 step = 0.5，使機器人以較快的速度移動。
  - 手臂擺動：設定手臂擺動的角度和速度。擺動角度（swing_angle)：控制手臂和腿部的最大擺動角度，例如步行時為 40 度，跑步時為 70 度。
    擺動速度（swing)：設定手臂和腿部的擺動速度，步行時為 3，跑步時為 10。
    控制擺動方向：根據步伐變化控制擺動方向，使手臂和腿部交替擺動。
    當手臂或腿部擺動角度達到最大值（例如 swing_angle）時，反向擺動。
    利用 dir 變數來記錄當前擺動方向，例如當 dir = 1 時表示順時針擺動，dir = -1 時表示逆時針擺動。
  - 碰撞檢測：在每次移動前，檢查機器人下一步位置是否與障礙物重疊。
    預測下一步的位置 next_x 和 next_z。
    使用碰撞檢測函數 check_collision() 判斷下一步是否會撞到障礙物。
    若發生碰撞則停止移動，若無碰撞則更新機器人的位置。
- 跳躍

- 上升過程：

  使用迴圈逐步增加機器人位置的 Y 軸座標來模擬跳躍的上升過程。

  每次增加 Y 軸高度，例如 position[1] += 1，並繪製畫面，模擬逐漸跳起的效果。

  可以在多次回圈中控制跳起的高度，例如迴圈進行 5 次，每次增加 1，達到頂點。

- 頂點：

  在達到預定高度（跳躍頂點）後，停止上升。此時可以在頂點短暫停留，給觀察者一個跳到最高點的效果。

- 下降過程：

  使用迴圈逐步減少 Y 軸高度，以模擬機器人從頂點下降的過程。

  每次降低 Y 軸的值，例如 position[1] -= 1，直到回到原始地面高度。

- 蹲下
  - 調整膝蓋角度：

    將膝蓋關節的角度設置為彎曲，例如 rknee_joint_angle = -70 和 lknee_joint_angle = -70。

  - 降低整體位置：

    調整機器人的 Y 軸位置，使整體位置降低，模擬重心下降的效果。

    例如，position[1] -= 1.5 * sin(40)，使機器人隨著膝蓋彎曲逐漸降低身體高度。

- 揮手
  - 手臂擺動：

    使用肩膀關節的旋轉來控制手臂的擺動，例如 rhand_joint_angle[0] += 18。

    每次迴圈更新肩膀的角度，使手臂逐步抬起至一定角度，模擬開始揮手的動作。

  - 肘部擺動：

    使用肘部關節的旋轉來控制手臂的揮動，例如 relbow_joint_angle[0] += 9。

    逐步增加肘部的角度，並將肘部向上擺動至最大角度，然後再逐步減少角度以恢復原位。

- 舞蹈
  - 手臂擺動：

    設定手臂擺動的動作步驟，使左右手臂交替擺動。

    例如，rhand_joint_angle[0] += 18 和 lhand_joint_angle[0] -= 18，使手臂朝相反方向擺動，並逐步增加擺動幅度。

- 膝蓋彎曲與蹲下：。

  使用 squat() 函數將膝蓋彎曲，同時降低機器人的高度。

- 重複舞蹈步驟：

  將各個動作以迴圈方式重複多次，使機器人連續執行多組舞步。

  在每個動作間加入短暫的停頓，例如 Sleep(100)。

- 發射子彈
  - 子彈結構體定義

    x, y, z: 子彈在三維空間中的位置。

    dx, dz: 子彈的方向矢量，用來控制子彈運動的方向和速度。

  - 生成子彈

    shoot_bullet() 函數: 當用戶按下發射鍵時（空白鍵），調用此函數發射子彈。

    生成新子彈並設置其初始位置為機器人當前的位置。

    設定子彈的運動方向 dx 和 dz，以機器人的當前方向 self_ang 來決定子彈的移動方向。

    將新子彈加入到子彈 vector 中。

  - 子彈移動

    update_bullets() 函數: 每次刷新畫面時，根據子彈的方向和速度更新子彈的位置。

    對每顆子彈，更新其位置，並重新繪製在新位置上。

    當子彈超出指定的範圍（x 或 z 坐標超過 100）時，從子彈 vector 中刪除。

  - 設置定時器自動刷新

    定時刷新畫面: 通過 glutTimerFunc 來設置畫面自動刷新，更新子彈的位置。

    timer() 函數每隔 16 毫秒刷新一次畫面。

    在 timer() 函數中調用 glutPostRedisplay()，從而自動重繪畫面。

    在 main 中啟動定時器