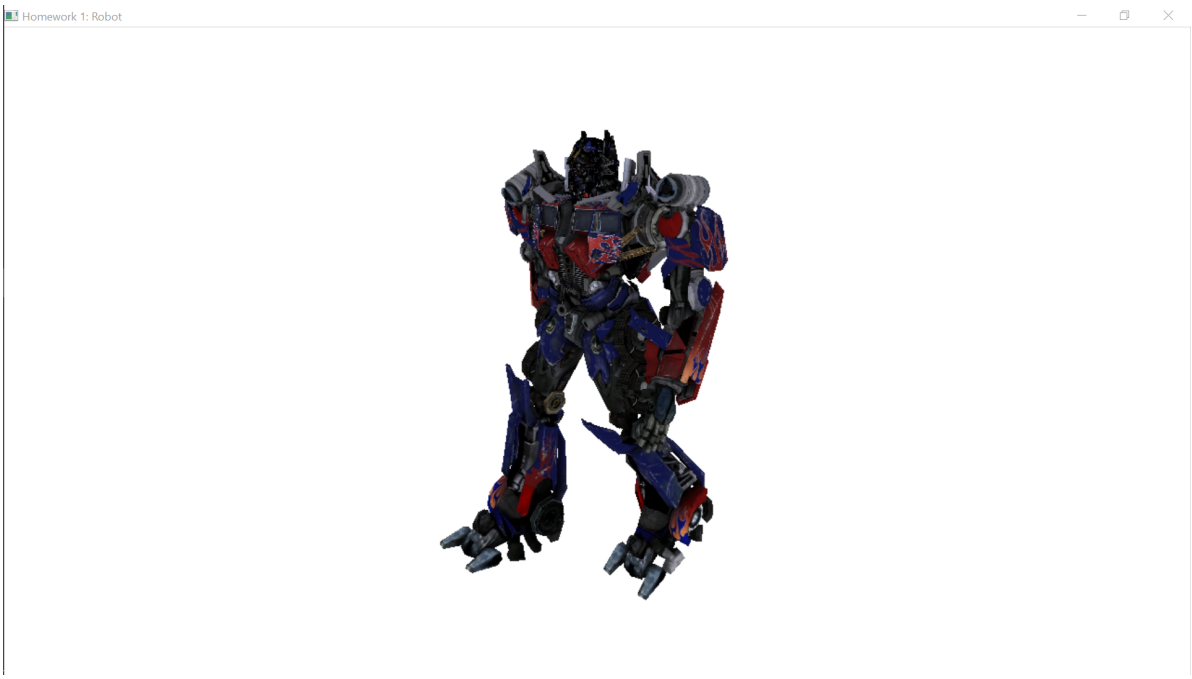
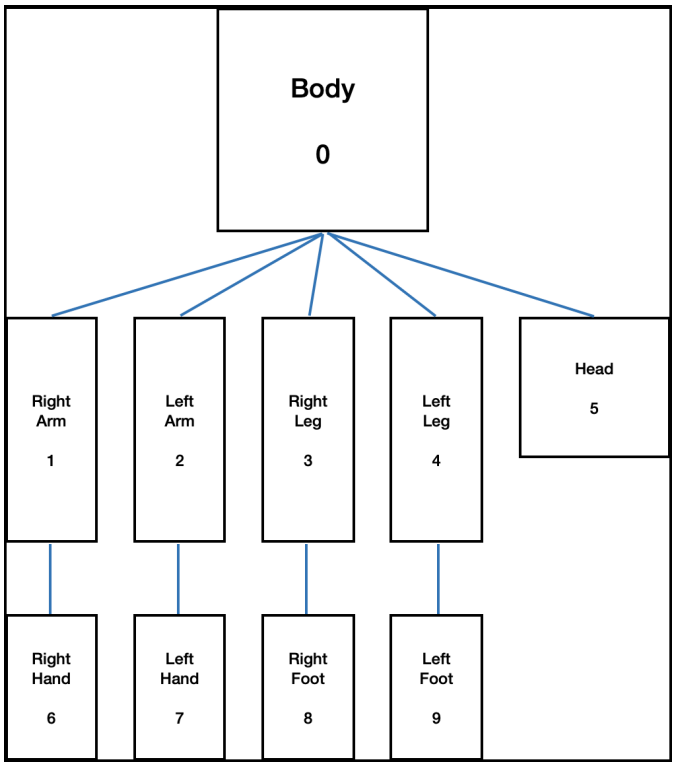
GPA Assignment 1

106034026 陳彥如

Sceenshot



Relationship



How to use

- 應用程式打開後，柯博文(我的機器人)便會開始行走
- 使用鍵盤可以操控機器人移動

    - `Q`: 向上
    - `E`: 向下
    - `W`: 向後
    - `S`: 向前
    - `A`: 向左
    - `D`: 向右
- 使用鍵盤可以操控機器人行走的速度
    - `↑`: 速度增加
    - `↓`: 速度減少
- 使用滑鼠左鍵可以操控機器人的旋轉
    - 按住 `滑鼠左鍵` 機器人便會沿著逆時針方向旋轉
    - 放開 `滑鼠左鍵` 機器人便會停止轉動
- 使用滑鼠右鍵可以叫出Menu

    - `Timer` 的 `Start` / `Stop` 可以操控動畫的播放/暫停
    - `Action` 可以選擇動畫種類
        - `Walking` 機器人走動
        - `Default` 機器人沿著Y軸旋轉

IDE version

Visual Studio 2019

Functions

透過更改助教所提供的 `Quiz Framework` 來製作本次的作業

- 使用Linked List來將每一個零件根據上述的Relationship進行連結
- 並將物件的原點存好，(是根據parents的相對關係)

```
void My_framework()
{
    m_shapes[0].parent = NULL;
    for (int i = 1; i <= 5; i++) {
        m_shapes[i].parent = &m_shapes[0];
    }
    for (int i = 6; i < 10; i++) {
        m_shapes[i].parent = &m_shapes[i - 5];
    }
    m_shapes[0].center = vec3(0, 0, 0); // body
    m_shapes[1].center = vec3(-5.33288, 83.0472, -172.447); // rightArm
    m_shapes[2].center = vec3(-5.33288, 83.0472, 172.444); // leftArm
    m_shapes[3].center = vec3(19.6564, -114.218, -57.4393); // rightLeg
    m_shapes[4].center = vec3(19.6564, -114.218, 57.4393); // leftLeg
    m_shapes[5].center = vec3(10.3625, 136.248, 0.587848); // head
    m_shapes[6].center = vec3(-25.7367, -50.9645, -215.277); //
rightHand
```

```
    m_shapes[7].center = vec3(-25.7367, -50.9645, 215.277); // leftHand
    m_shapes[8].center = vec3(32.2727, -280.651, -82.3083); //
rightFoot
    m_shapes[9].center = vec3(32.2727, -280.651, 82.3083); // leftFoot

    for (int i = 1; i < 10; i++) {
        m_shapes[i].center = m_shapes[i].center - m_shapes[i].parent-
>center;
    }

    std::cout << "My_framework() DONE" << endl;
}
```

- 根據不同的物件，設定rotation_matrix

```
mat4 My_Model(int index, int state)
{
    mat4 my_model(1.0f);
    mat4 translation_matrix = translate(mat4(1.0f),
m_shapes[index].center);
    mat4 move_matrix = translate(mat4(1.0f), temp);
    vec3 rotate_axis = vec3();
    mat4 rotation_matrix(1.0f);
    mat4 scale_matrix = scale(mat4(1.0f), vec3(0.01, 0.01, 0.01));

    switch (state) {
        case STATE_WALK:
        {
            rotate_axis = vec3(0.0, 0.0, 1.0);
            switch (index) {
                case 0: // body
                {
                    rotate_axis = vec3(0.0, 1.0, 0.0);
                    rotation_matrix = rotate(mat4(1.0f),
radians(mouseAngle), rotate_axis);
                }
                    break;
                case 1: // rightArm
                {
                    float angle = sin(freq * timer_cnt) * 30;
                    rotation_matrix = rotate(mat4(1.0f),
radians(angle), rotate_axis);
                }
                    break;
                    .
                    .
                    .
            }
            .
            .
            .
            m_shapes[index].nodeModel = my_model;
        }
    }
}
```

- 將物件的model回傳到 `My_Display()` function 中
  每一個object的model除了自己的`translation_matrix`、`rotation_matrix`外，還需乘上parent的model

```
mat4 My_Model(int index, int state)
{
    .
    .
    .

    if (m_shapes[index].parent != NULL) {
                my_model = m_shapes[index].parent->nodeModel *
translation_matrix * rotation_matrix;
            }
            else {
                my_model = move_matrix * scale_matrix *
translation_matrix * rotation_matrix;
            }
            m_shapes[index].nodeModel = my_model;
    .
    .
    .
}
```